Платформа видеоконференций ВКонтакте — сделано удалённо

Александр Тоболь

Технический директор ВКонтакте и единой платформы звонков Mail.ru Group





Почему про видеоконференции?

Covid-19 / Zoom-a-Zoom-Zoom

BBOViz Covid-19

There's only one winner in the work-from-home meeting app wars

- Zoom
- Microsoft Teams
- Google Meet
- Houseparty
- Skype for iPhone
- Discord
- Google Duo
- Google Hangouts
- WeChat Work
- Skype for Business
- Marco Polo - Stay In Touch
- Slack

Daily worldwide app downloads, in millions

3M
2.5
2
1.5
1
0.5
0

March 2020 | Pandemic declared | April

Source: Apptopia

BEN SCHOTT

BloombergOpinion



Групповые звонки и голосовые чаты в 2020/2021

- Как устроены Zoom / Clubhouse?
- Что мне взять для моего сервиса / стартапа?































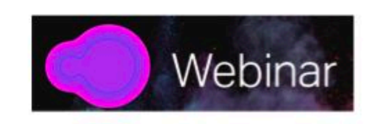






















Как мир видеоконференций менялся во время пандемии

Март 2020	**Discord** увеличил кол-во участников видеочата до 50
Апрель 2020	**WhatsApp** увеличил кол-во участников до 8
	Facebook Messanger запустил Desktop app
	Clubhouse iOS app
	Бесплатная версия **Google Meet**
Май 2020	**Zoom** отменил ограничение в 40 минут
Июль 2020	Видеочаты в **Discord mobile app**
Август 2020	**Facebook** добавил Zoom, Cisco, BJ, … в **Facebook Portal**
Сентябрь 2020	Шумоподавление в **Google Meet**
Январь 2021	Илон Маск в **Clubhouse**
Апрель 2021	**Facebook Hotline**
Март 2021	Групповые видеозвонки на **WhatsApp Desktop**
Май 2021	Групповые видеозвонки **Telegram**



Звонки ВКонтакте

2012	Видеозвонки на вебе
2018	Мобильные видеозвонки один на один
Май 2020	Групповые видеозвонки на **8 человек**
Сентябрь 2020	Групповые видеозвонки на **128 человек**
2021	Цель — **1000+ человек**



Что нас сейчас ждёт?



- NACK, FEC, PLC
- AEC, NS, VAD, AGC
- MOS, PESQ, NISKA
- MCU, SFU, SVC, Simulcast
- DTX, FIR, I|P|B - frames



План доклада

- Теория
- Практическая часть на примере звонков ВКонтакте
- Подходы, позволяющие делать конференции на 1000+ и более
- Анализ подходов крупных сервисов
- Список trick'ов для видеоконференций
- Open-source-решения
- ML в звонках
- Общий подход к созданию технически сложных высоконагруженных продуктов



Disclaimer

Выводы и заключения по конкурентам сделаны исключительно из общедоступной информации, общих наблюдений и измерений и могут быть ошибочными.



Требования, или Чего хотят пользователи

Какие интернет-звонки хотят пользователи?

01 **Качественные**

02 С видео

03 Безопасные

04 Без ограничений по времени

05 Доступные на всех устройствах

И, КОНЕЧНО, ВСЁ ЭТО БЕСПЛАТНО!



Что пользователи не любят больше всего

- **Плохое качество аудио или видео**
- **Нестабильное соединение**
- **Если часто вылетает или зависает**
- Сложный интерфейс
- Нет записи
- **Греется телефон**
- **Некомфортно общаться**
- Нет или не нравятся маски (для личного общения)



Функционал конкурентов

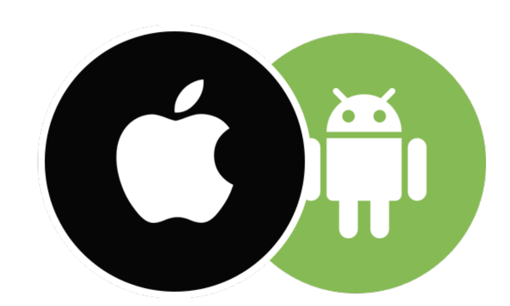
	Clubhouse	WhatsApp	Viber	Fb	Discord	Skype	Zoom	Google Meet
Число участников	10K	8	30	50	50	50	100 ($ —1000)	100 ($ —250)
Платформы						*	*	
Подключение по ссылке	✓	✗	✓	✓	✗	✓	✓	✓
Демонстрация экрана	✗	✗	✓	✓	High FPS	✓	**High Res**	High FPS
Запись	✗	✗	✗	✗	✗	✓	локально	$$
Виртуальные фоны и маски	✗	✗	✗	✓	✗	✓	✓	✓

- тут серебряной пули нет



Требования к звонкам ВКонтакте

- Неограниченное кол-во участников *
- Работа на платформах






- Низкое потребление серверных ресурсов
- Высокое качество звонков
- Стабильность звонков
- Низкое потребление ресурсов клиента

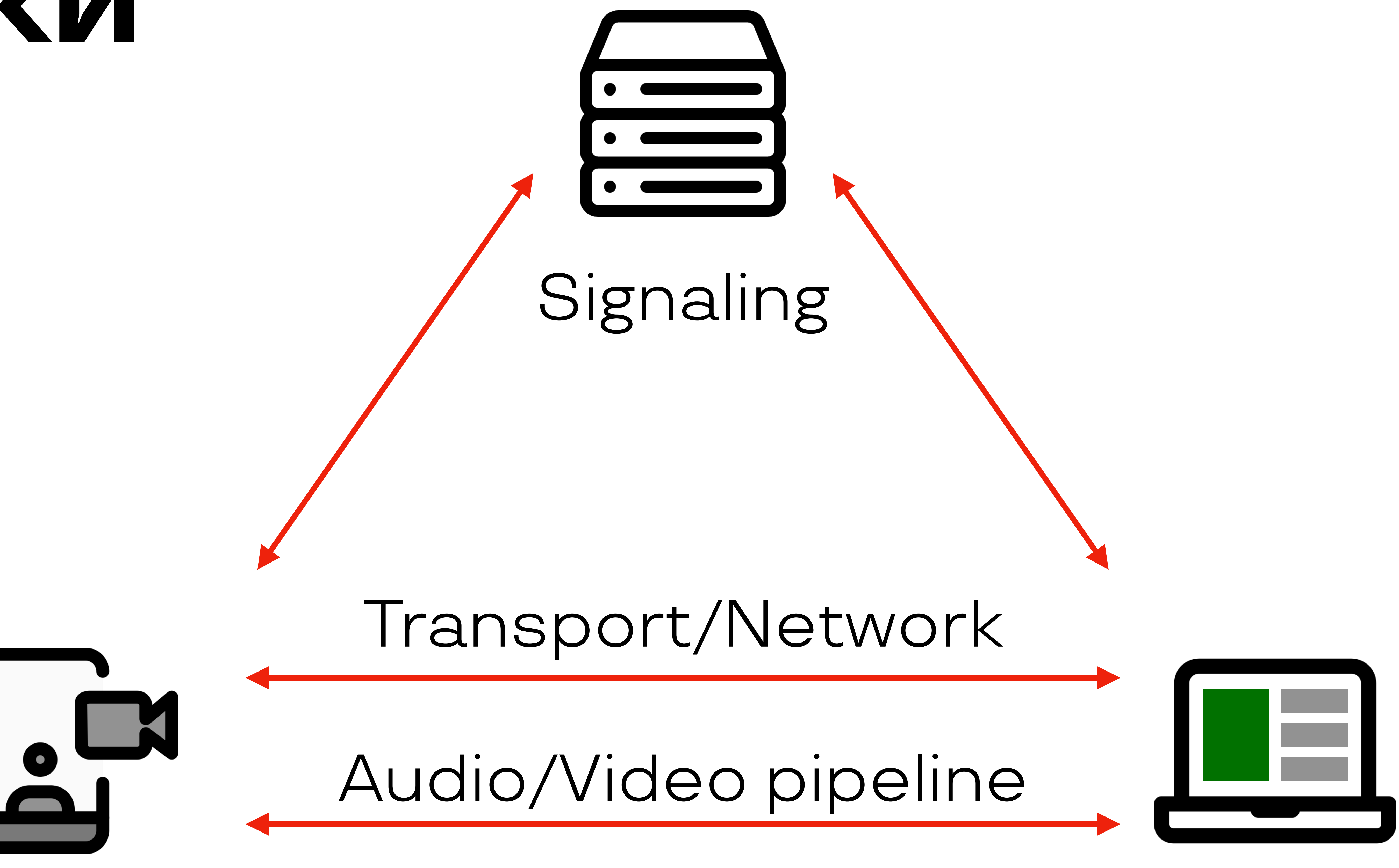
СДЕЛАТЬ КРУТЫЕ ЗВОНКИ!





Теория

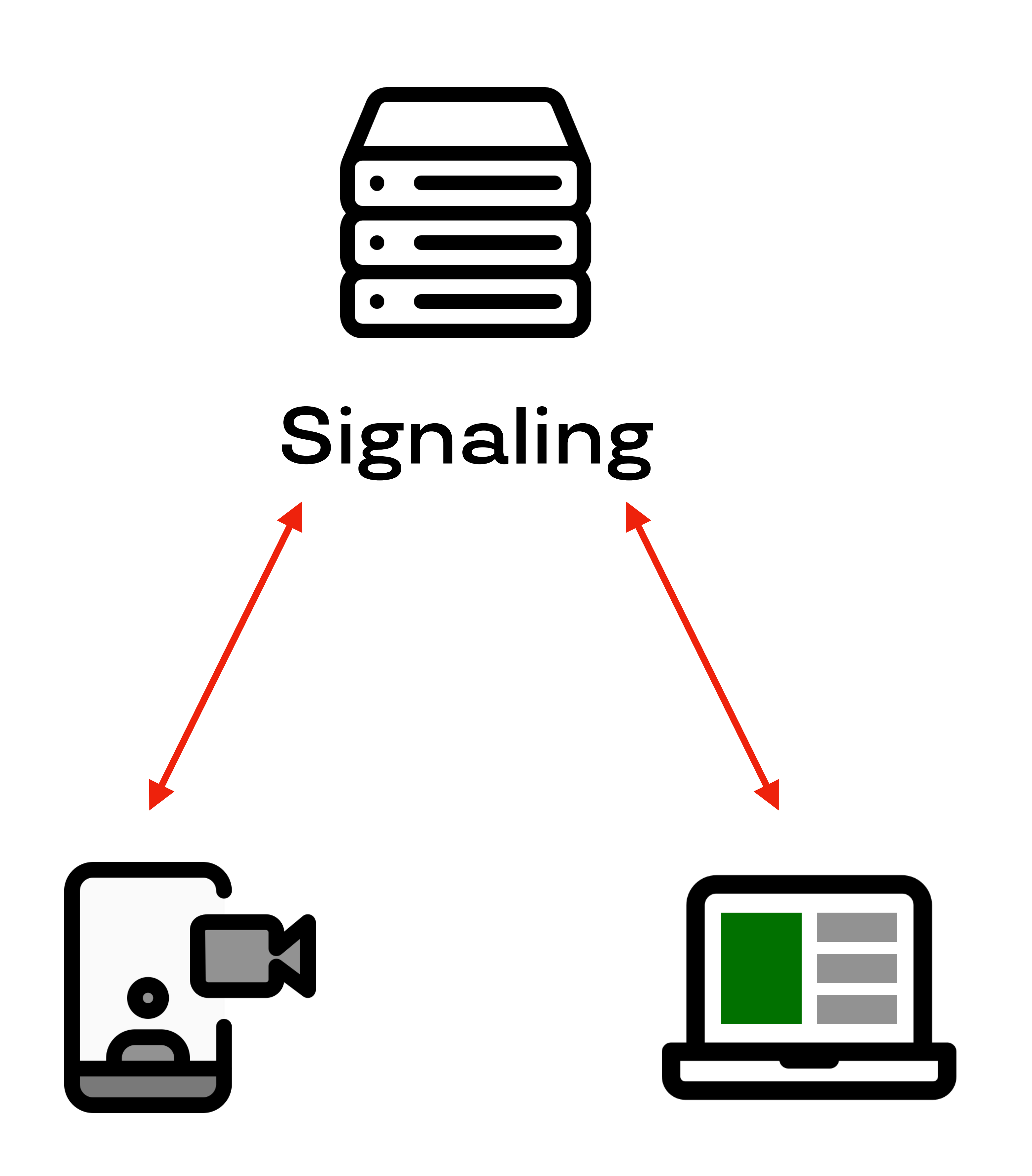
Как устроены любые звонки





Уровень логики — Signaling (координация участников)

- аутентификация / авторизация
- распространение сообщений между участниками с гарантией порядка и подтверждением
- установка сетевого соединения с сервером или между участниками
- список участников
- блокировки
- зал ожидания...

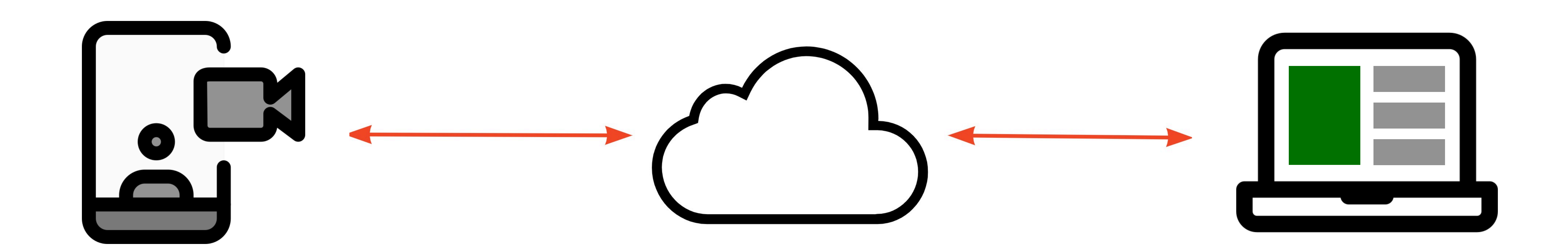


* похож на мессенджер через WebSocket



Теория: сетевой уровень

Сетевой уровень



- минимальные задержки
- приоритет на UDP



Характеристики сети

Server

1 2 3 ... M N

RTT

Client

1 2 3 ... M

PL

BW = Packets x Size / sec

- Bandwidth – Mbps
- RTT – ms
- Packet loss – %



RTT — случайная величина

ГДЕ АРГУМЕНТЫ?

```
[MacBook-Pro:~ aleksander.tobol$ ping highload.ru
PING highload.ru (178.248.233.16): 56 data bytes
64 bytes from 178.248.233.16: icmp_seq=0 ttl=51 time=12.607 ms
64 bytes from 178.248.233.16: icmp_seq=1 ttl=51 time=58.139 ms
64 bytes from 178.248.233.16: icmp_seq=2 ttl=51 time=100.236 ms
64 bytes from 178.248.233.16: icmp_seq=3 ttl=51 time=148.689 ms
```

350
300
250
200
150
100
50
0
1 3 5 7 9 11 13 15 17 19 21 23 25 27 29 31 33 35 37 39 41 43 45 47 49 51 53 55 57 59 61 63 65 67 69 71 73 75 77 79 81 83 85 87 89 91 93 95 97 99



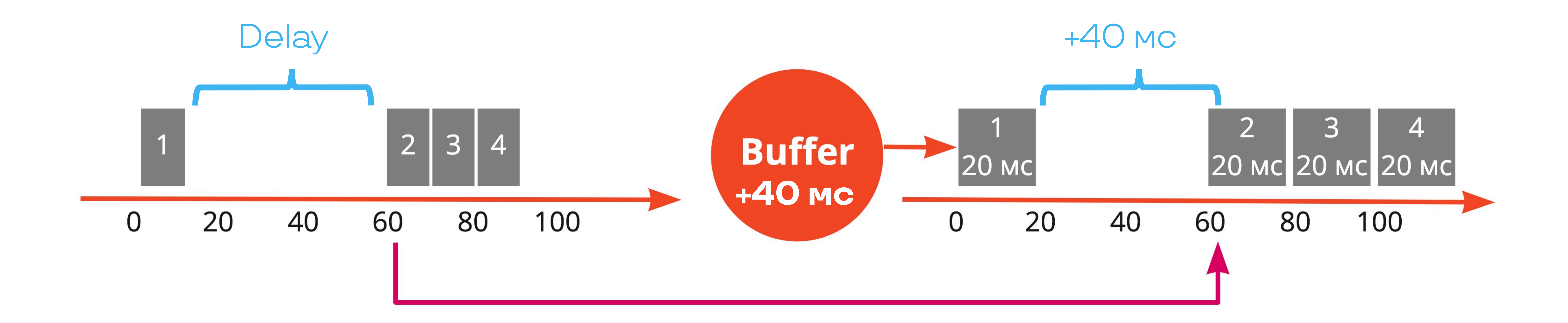
Jitter — оценка изменения задержки пакетов

Jitter = (maxDelay – minDelay)

есть другой способ измерения в RTP из rfc 3550



Борьба с jitter'ом — buffer





Борьба с Packet loss — FEC

1 2 3 4 P 5

K

P

P = XOR(K)

Network

1 2 4 P 5

3

3 = XOR(K-1,P)

- overhead даже, если нет потерь
- чиним фиксированный % потерь



Борьба с Packet loss — FEC + NACK

Client

Server

pkt 1

pkt 2

pkt3

packet drop

pkt 4

packet drop

pkt5

pkt 3 - FEC

nack 4

pkt 4



Борьба с RTT — CDN

* peering операторов

Санкт-Петербург
Екатеринбург
Красноярск
Петропавловск-Камчатский
Новосибирск
Челябинск
Москва
Минск
Самара
Акколь
Хабаровск
Франкфурт
Ростов-на-Дону
Молдова
Тараз
Грузия
Шымкент
Крым
Азербайджан
Киргизия
Армения



Метрики сети

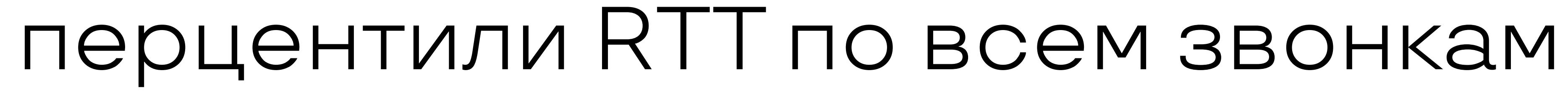
Как оценить RTT по всем пользователям?

99p

Медиана

0,5	
0,75	
0,98	
0,99	
0,999	

600
500
400
300
200
100
0





Какая у нас сеть?

	round-trip time	bandwidth	packet loss	packet jitter / 10s
p50	140 мс	3.3 Мбит/с	0.9 %	60 мс
p99	550 мс	0.4 Мбит/с	2.2 %	1400 мс

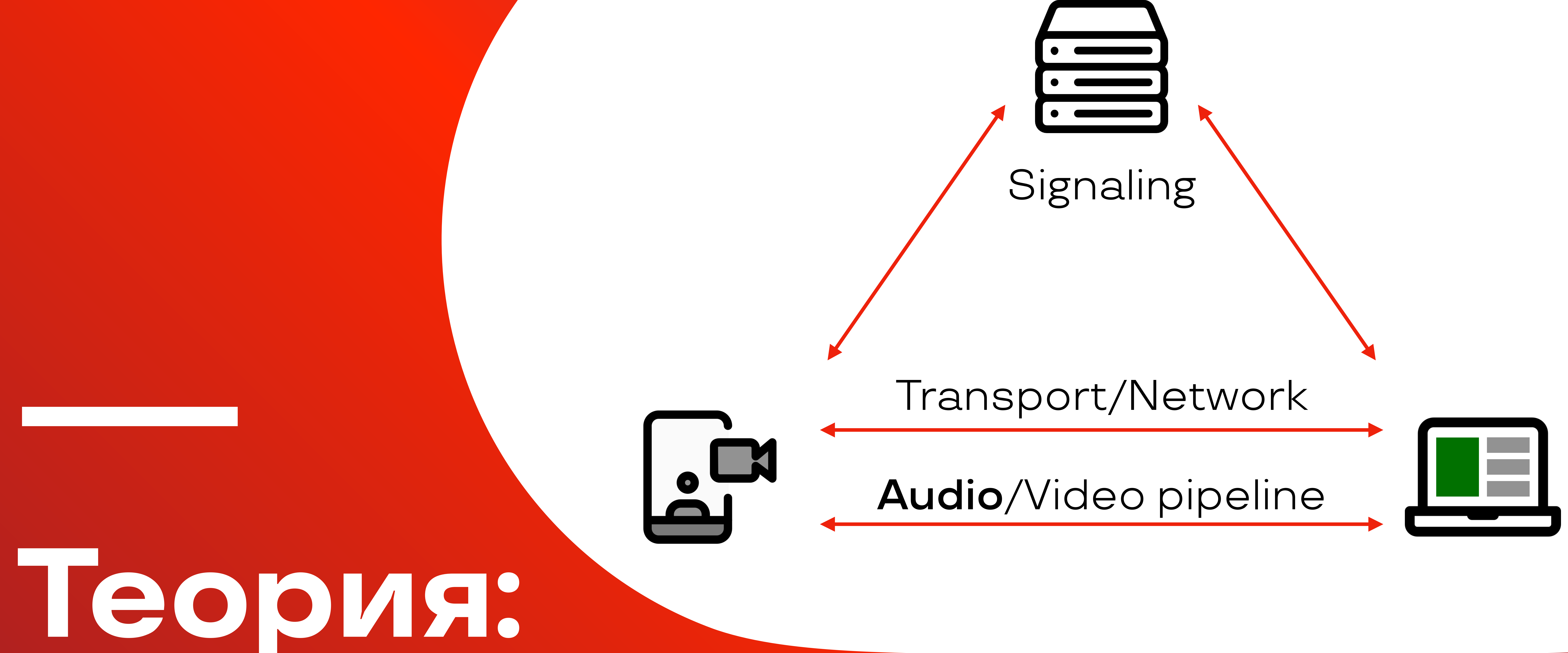


Сетевой уровень — итого



- Сеть определяется переменными величинами: Bandwidth/RTT/Packet loss
- В зависимости от geo ваших пользователей и CDN у вас будут свои цифры
- Одно и то же решение для звонков, развернутое в разных условиях, может показывать разные результаты
- Не смотрите на среднее, опирайтесь на медиану и перцентили
- Jitter компенсируется через jitter buffer и вызывает задержку
- RTT/BW можно улучшать за счет GEO CDN
- Packet loss чинится через FEC/NACK





Теория: audio pipeline

Audio pipeline

Алиса

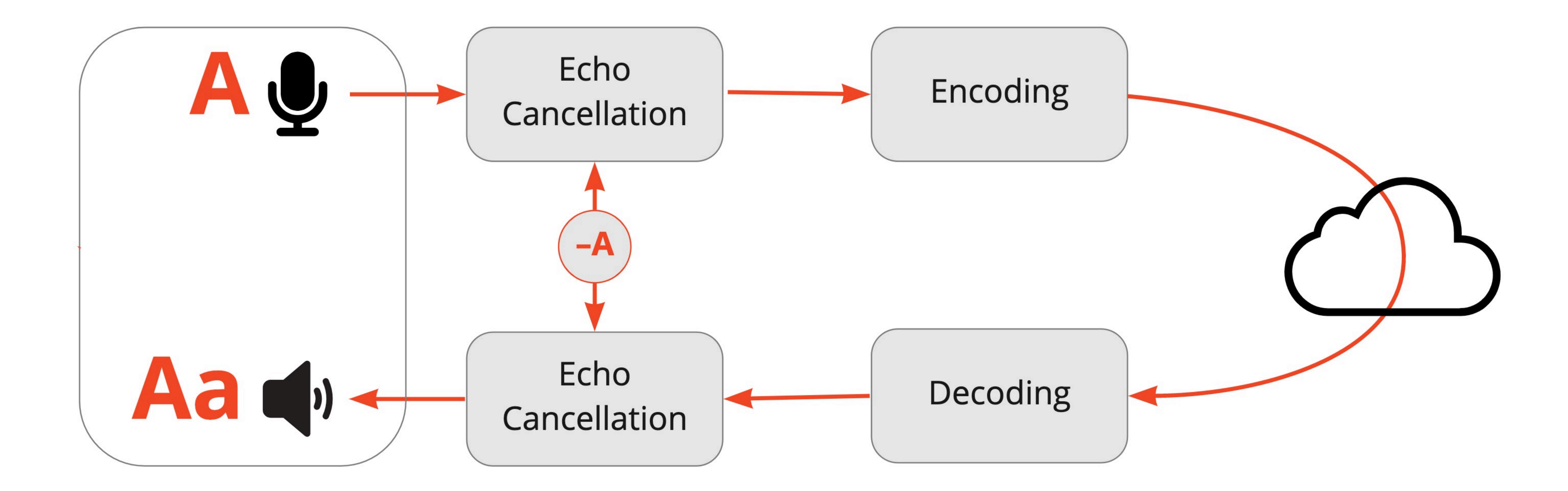
Echo Cancellation
Noise suppression
Voice activity detection
Automatic gain control
Encoding

Echo Cancellation
Audio mixer
Decoding
FEC
Jitter Buffer

Борис



Эхо


A
Echo Cancellation
Encoding
-A
Aa
Echo Cancellation
Decoding


HL
HighLoad++
Весна 2021

Audio pipeline

Алиса

Борис

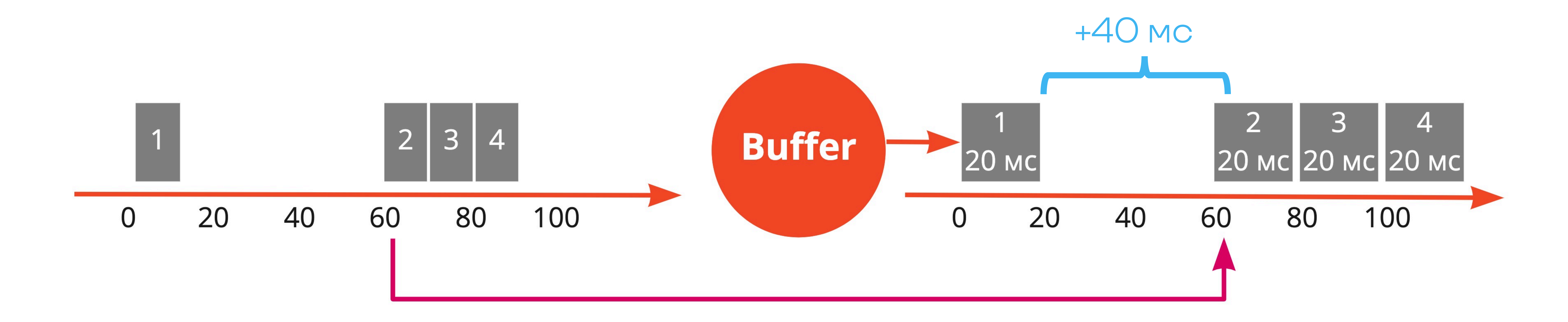
Echo Cancellation → Noise suppression → Voice activity detection → Automatic gain control → Encoding → ••• →

••• ← Jitter Buffer ← FEC ← Decoding ← Audio mixer ← Echo Cancellation ←

time stretching



Компенсация задержки



- JB ускоряет тишину
- Чем лучше VAD, тем больше можно догнать без искажений
- Sonic ускоряет голос https://github.com/waywardgeek/sonic





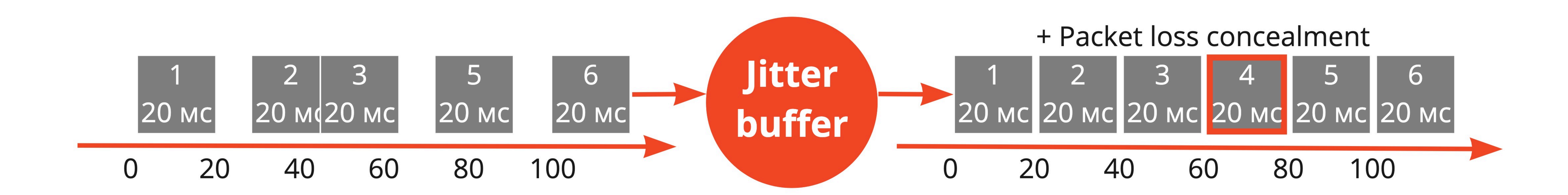
Audio codec

fullband stereo
fullband
super-wideband
wideband
narrowband
Quality
Opus
AAC
Vorbis
MP3
G.719
G.722.1C
AMR-WB
Speex
G.722
AMR-NB
iLBC
G.711
8
16
32
64
128
bitrate (kb/s)
● royalty-free, open-source

- OPUS де-факто



Маскировка потери пакетов



- OPUS PLC хорошо чинит один пакет ~ 20мс



НЕ путать



PLC (Packet Loss Concealment) — маскирует потери пакетов.



FEC — избыточное кодирование для исправления ошибок.



Time stretching

Input delay

delay

80
60
40
20
0

0 100 200 300

time

delay

80
60
40
20
0

PLC
Silence
Pessimism
Acceleration

0 100 200 300

time

- latency vs distortion tradeoff



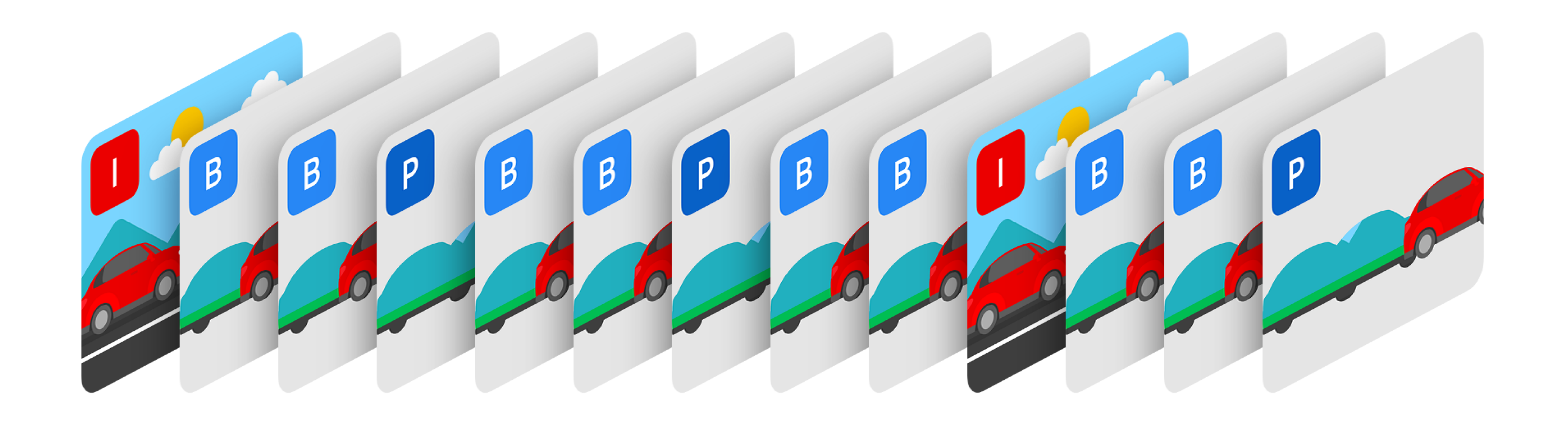
Итого

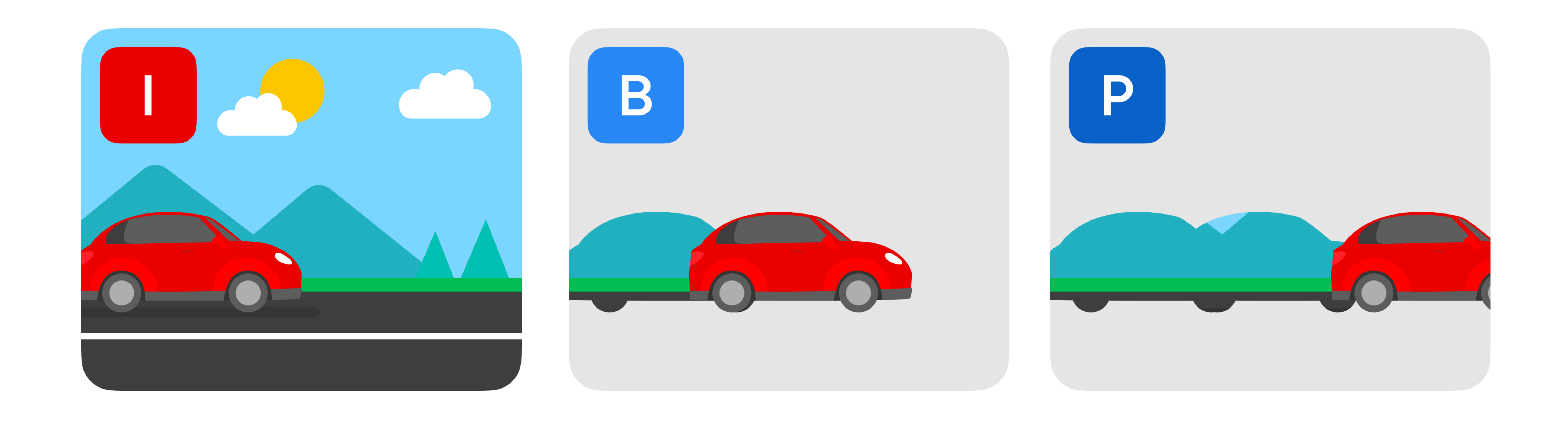
- AEC, NS, VAD, AGC
- Time-stretching
- JB умеет догонять на тишине
- Чем лучше VAD, тем больше можно догнать без искажений
- Tradeoff между latency и количеством искажений
- OPUS умеет PLC и FEC



Теория: video

Video codec







I-frame delay

I B B P B B P B B I B B P

P

- задержка секунды



FIR — Full Intra Request

I B B I B B P B B P B B I B B P

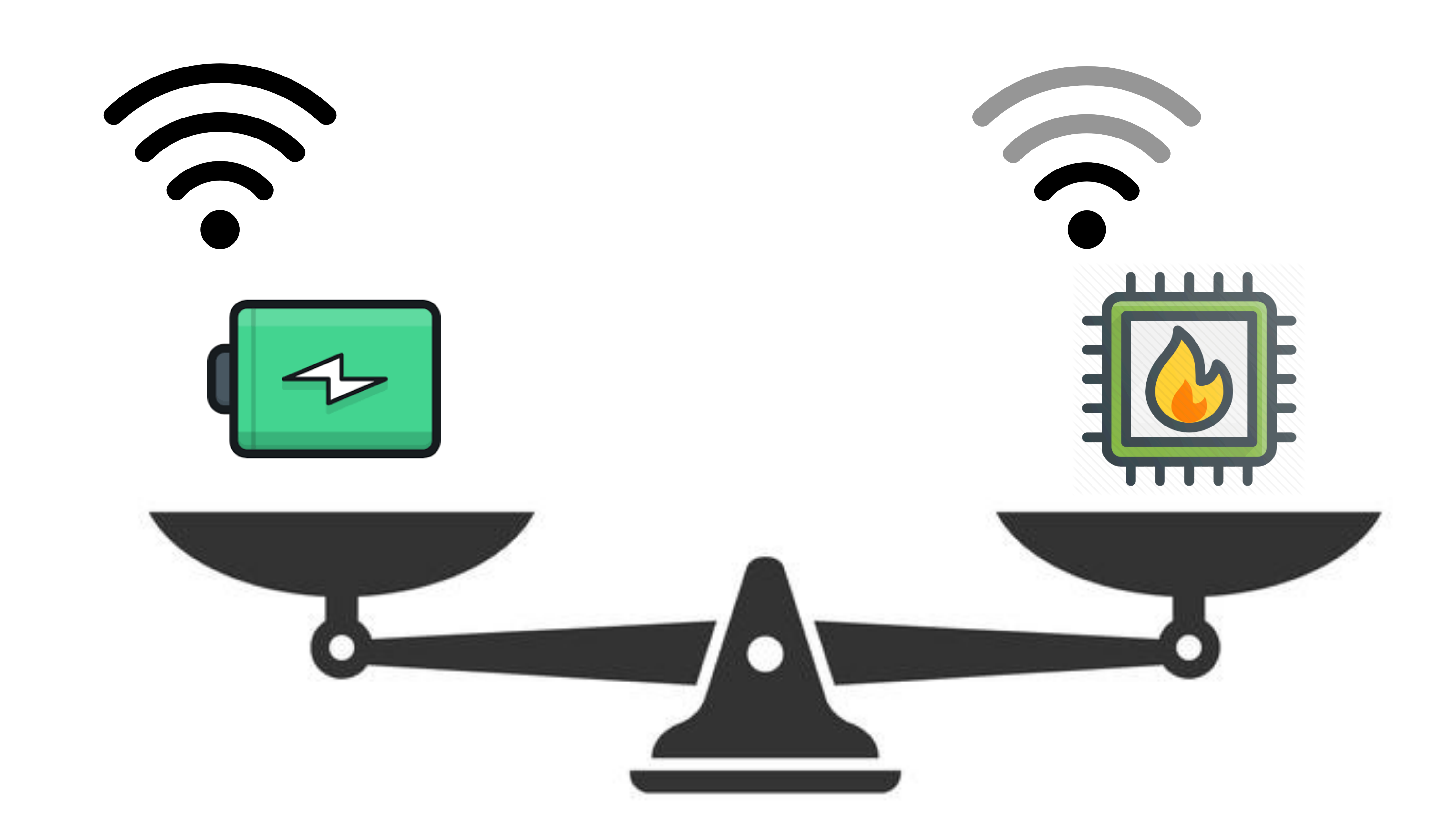
I

- НО лишний трафик



Codecs







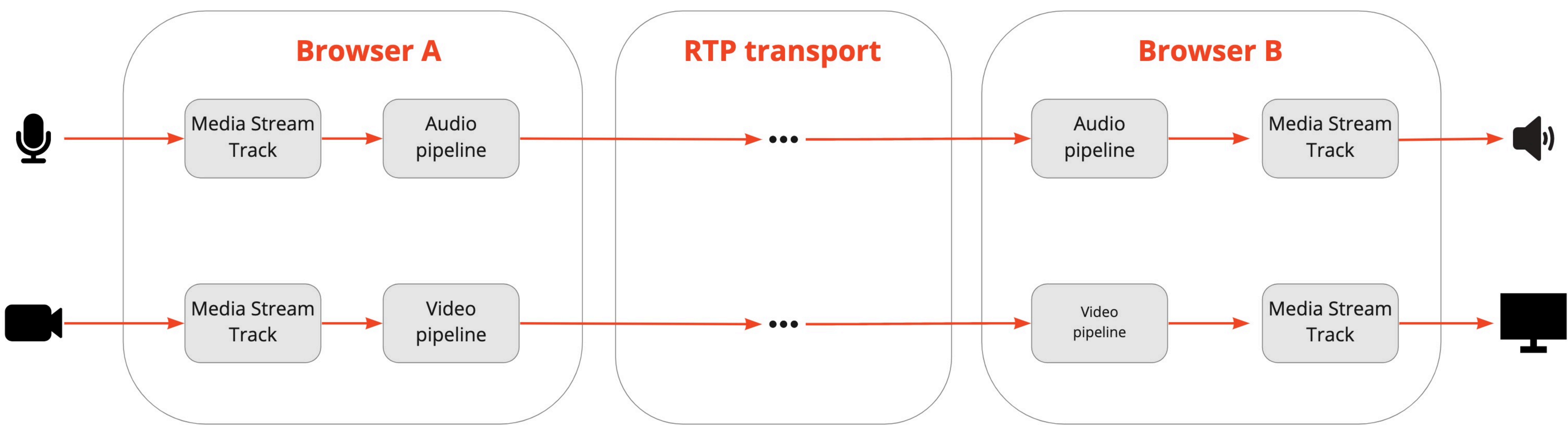
Итого

- I | P | B frames
- FIR — Full Intra Request
- Codec это tradeoff между CPU и Traffic



WebRTC — Web Real Time Communication





- Сетевой уровень
- Audio / video pipeline
- Единственное решение, работающе в web-браузерах
- Не реализует сигналинг и топологию



Весь pipeline звонка

- Сигналинг
- Установка транспортного соединения
- Обмен доступными кодеками
- Кодирование видео / аудио
- Отправка в сеть и адаптация
- Починка пропажи пакетов
- Компенсация задержек
- => видеоконференция



Оценка качества звонков

Какие интернет-звонки хотят пользователи?


КАЧЕСТВЕННЫЕ

ответили 80% респондентов





Оценка качества звонков



MOS — mean opinion score



- референсная метрика

Rating	Label
5	Excellent
4	Good
3	Fair
2	Poor
1	Bad

$$MOS = \frac{\sum_{n=1}^{N} R_n}{N}$$



PESQ MOS —
Perceptual Evaluation of Speech Quality

Original Source

Full Pipeline

Degradated Source

Subject

Model



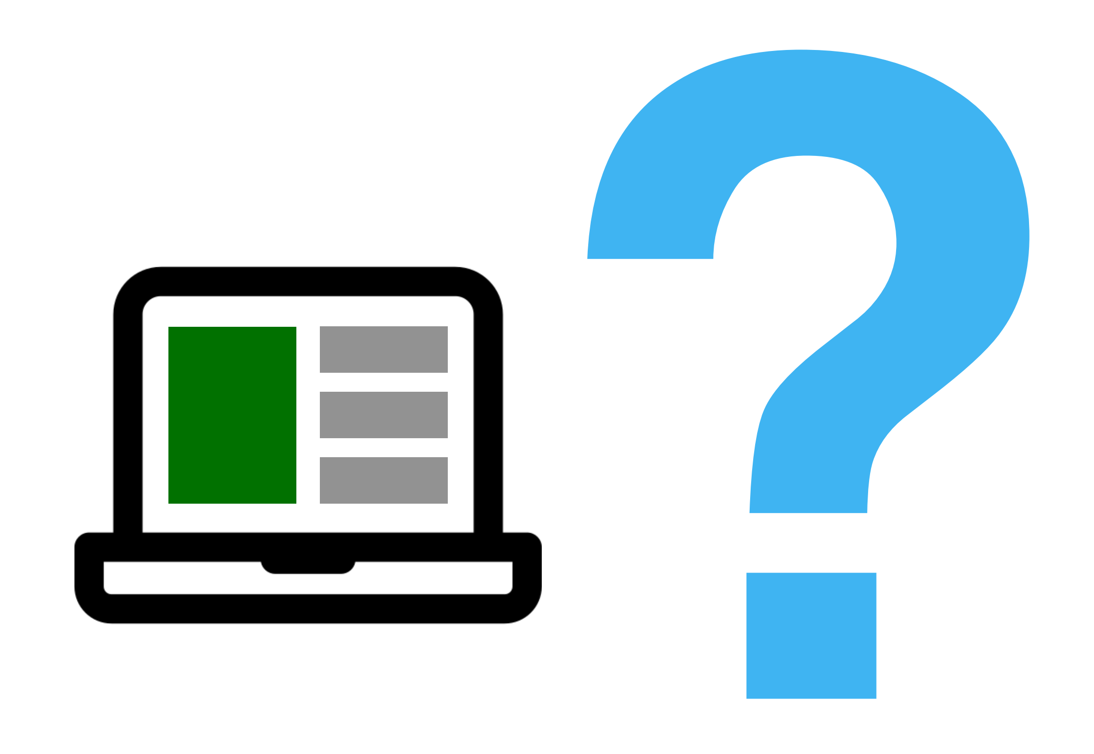
Метрики оценки качества звонков — NISQA

- **14,000** примеров с packet-loss, background noise и примеры из Zoom, Skype, WhatsApp
- **97,000** MOS рейтингов

% потерь	NISQA	PESQ MOS
5 %	4.3	3.2
10 %	3.4	2.4
15 %	1.7	1.8
20 %	1.4	1.5
25 %	1.2	1.2
50 %	0.9	1.0

- Считается на порядки быстрее POLQA и быстрее PESQ MOS
- Метрика работает на речевых сигналах — использовать после VAD’a.
- https://github.com/gabrielmittag/NISQA

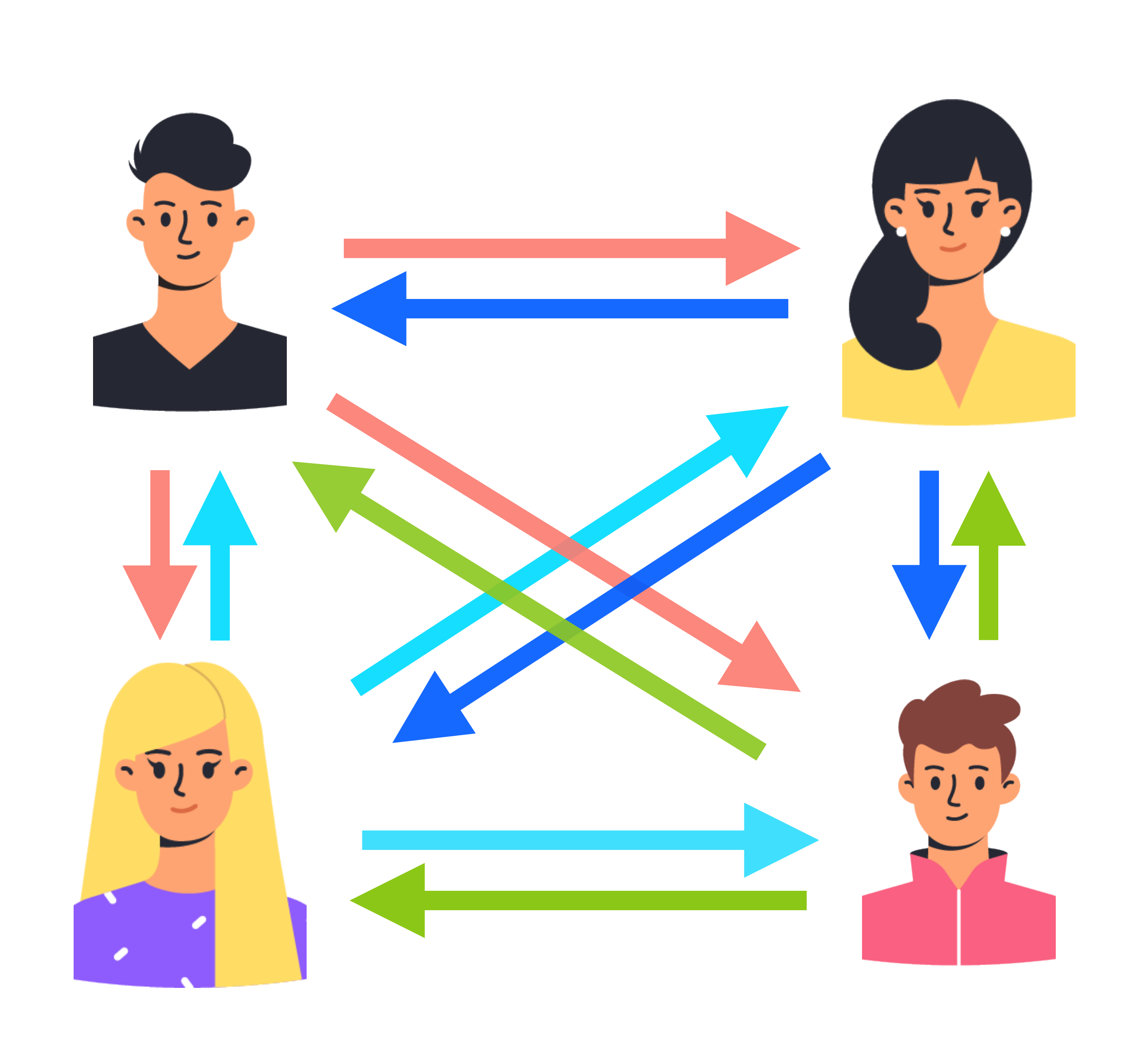


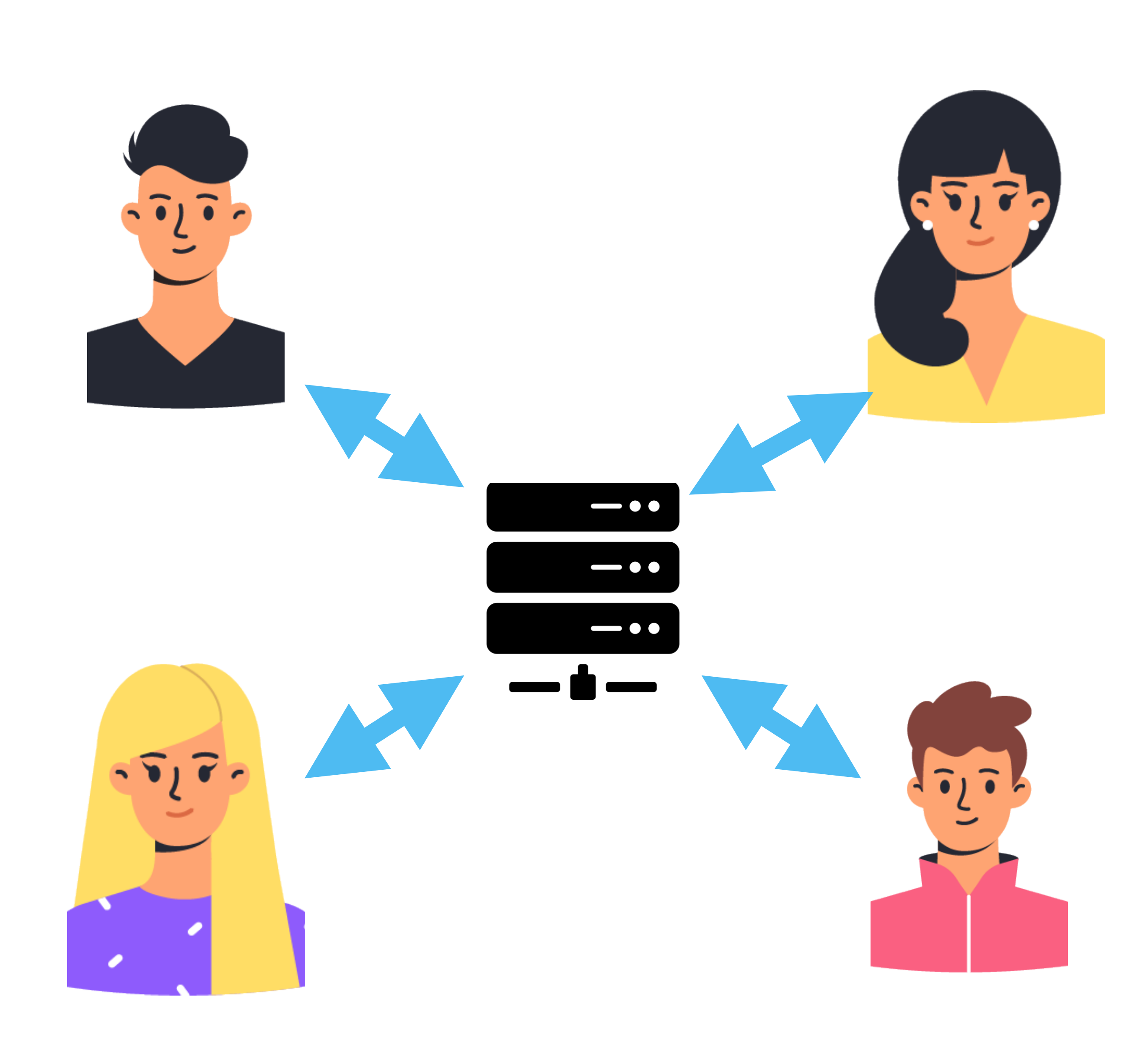


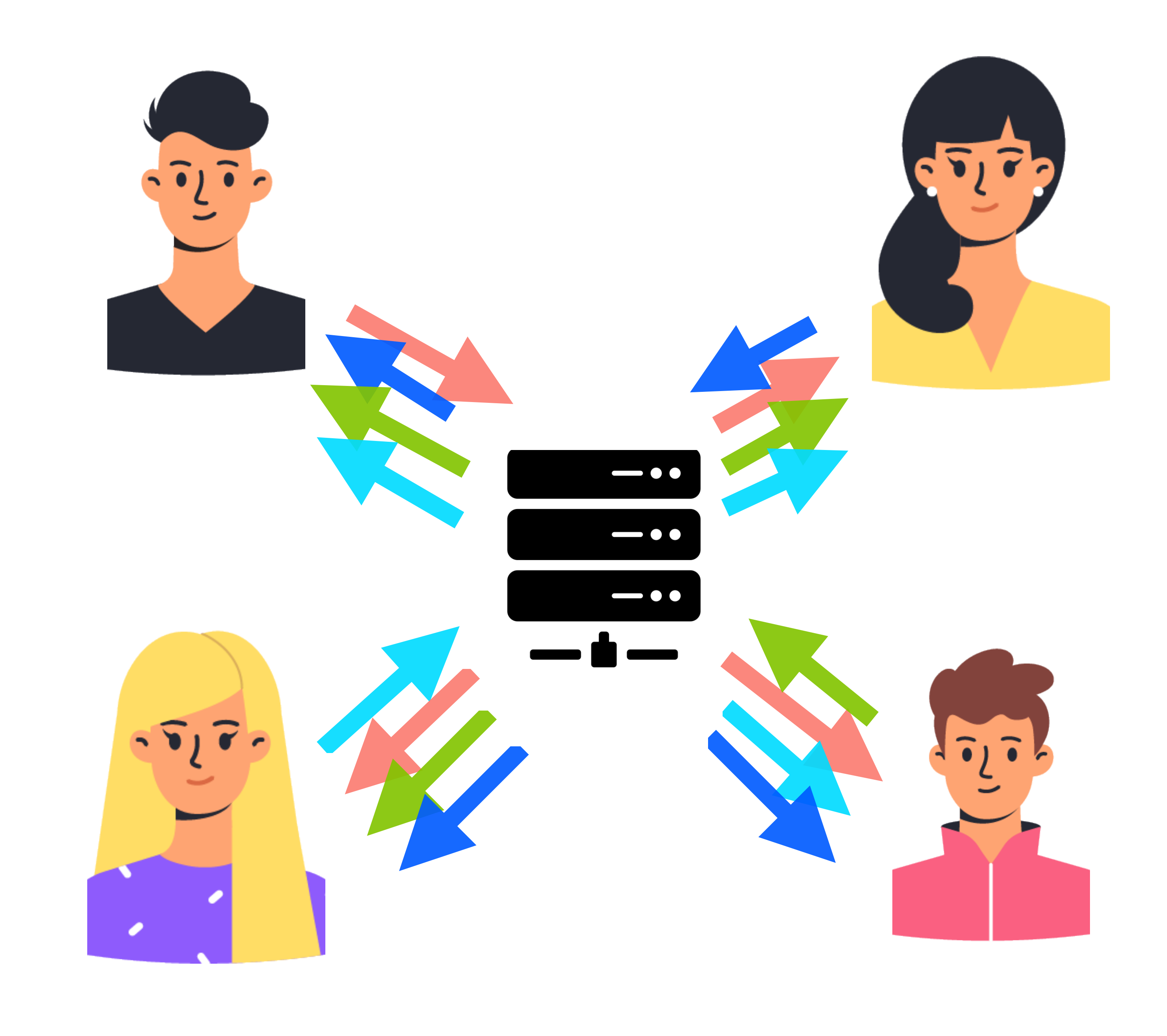


Групповые звонки и топологии

Как из p2p сделать групповые звонки







MESH

MCU — MIX

Multipoint
Conferencing Unit

SFU — FORWARDING

Selective
Forwarding Unit



Сравнение топологий на 4 участника

	MESH	MIX	FWD
in/out streams	3 / 3	1 / 1	3/ 1
input traffic	3 Мбит/сек	1 Мбит/сек	3 Мбит/сек
output traffic	3 Мбит/сек	1 Мбит/сек	1 Мбит/сек
client CPU	3 E + 3 D = 60%	1 E + 1 D = 20%	1 E + 3 D = 40%
server CPU	0	100 %	10 %
latency	min	max	avg
SIP / live	—	+	—
max participants	~ 8	∞	~ 50



Ошибки

Ошибки выбора топологий: full mesh > ~8





Serverless — forwarding broadcast





VOIP over TCP



Если PL=0% Jitter=0 BW=const



TCP против **UDP** или **будущее сетевых протоколов**
habr.com/ru/company/oleg-bunin/blog/461829/





Формальная постановка задачи

Качество

- min задержка
- min искажения
- min потребление ресурсов



Что влияет на качество

- Сеть пользователя:

  BW / RTT / PL / Jitter
- Устройство / браузер
- Доступные ресурсы на клиенте
- Количество участников в звонке

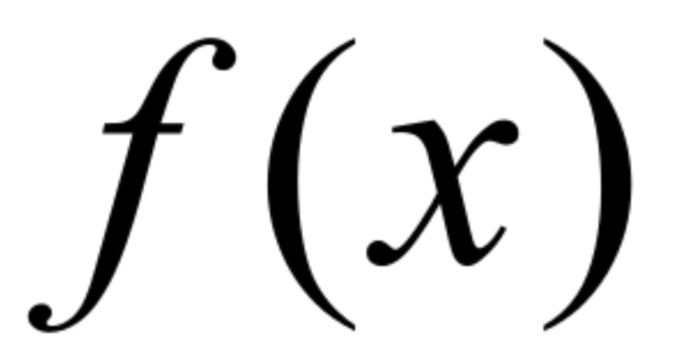


На что влияем мы

- Алгоритмы сетевого уровня
- Audio / Video pipeline
- Jitter buffer и алгоритмы компенсации задержки
- Audio / Video adaptation
- Топология групповых звонков



Формальная постановка задачи



Многокритериальная оптимизация:

$$\min_{X,V}\{\textbf{задержка}(X,V),\ \textbf{искажения}(X,V),\ \textbf{потребления ресурсов}(X,V)\},$$

где:

X = {BW, RTT, PL, Jitter, Device(CPU, OS, GPU, Resolution) | BROWSER, заряд батареи, перемещение в пространстве, количество участников, …}

V = {FEC, NACK, CDN, V/A ADAPTATION, TIME-STREATCHING ,CODECS, TOPOLOGY}

Для всего пространства X, ограниченного **99**-ым перцентилем

количество участников [1,1000+]



Практика
и Tricks

Tricks и зоны



Бери и
применяй



Анализ метрик
и тесты



Tradeoff’ы
и перцентили



Аудиотопология

SFU — audio forwarding

REJECTED

Microphone inputs

peaker outputs

input traffic	400 Кбит/сек
client CPU	1 E + 4 D = 50%
SIP / live	—
max participants	webRTC (max ~50 decoders)

* Без аудиомикса на клиенте нужно N–1 декодеров



MCU — audio mix

Microphone inputs

Server-side mix

+

–A

–B

–C

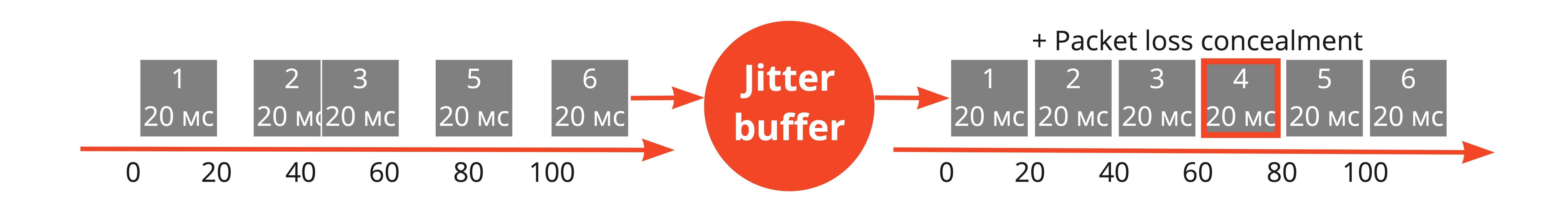
Speaker outputs

- вычитаем, иначе будет эхо



Jitter buffer и time stretching

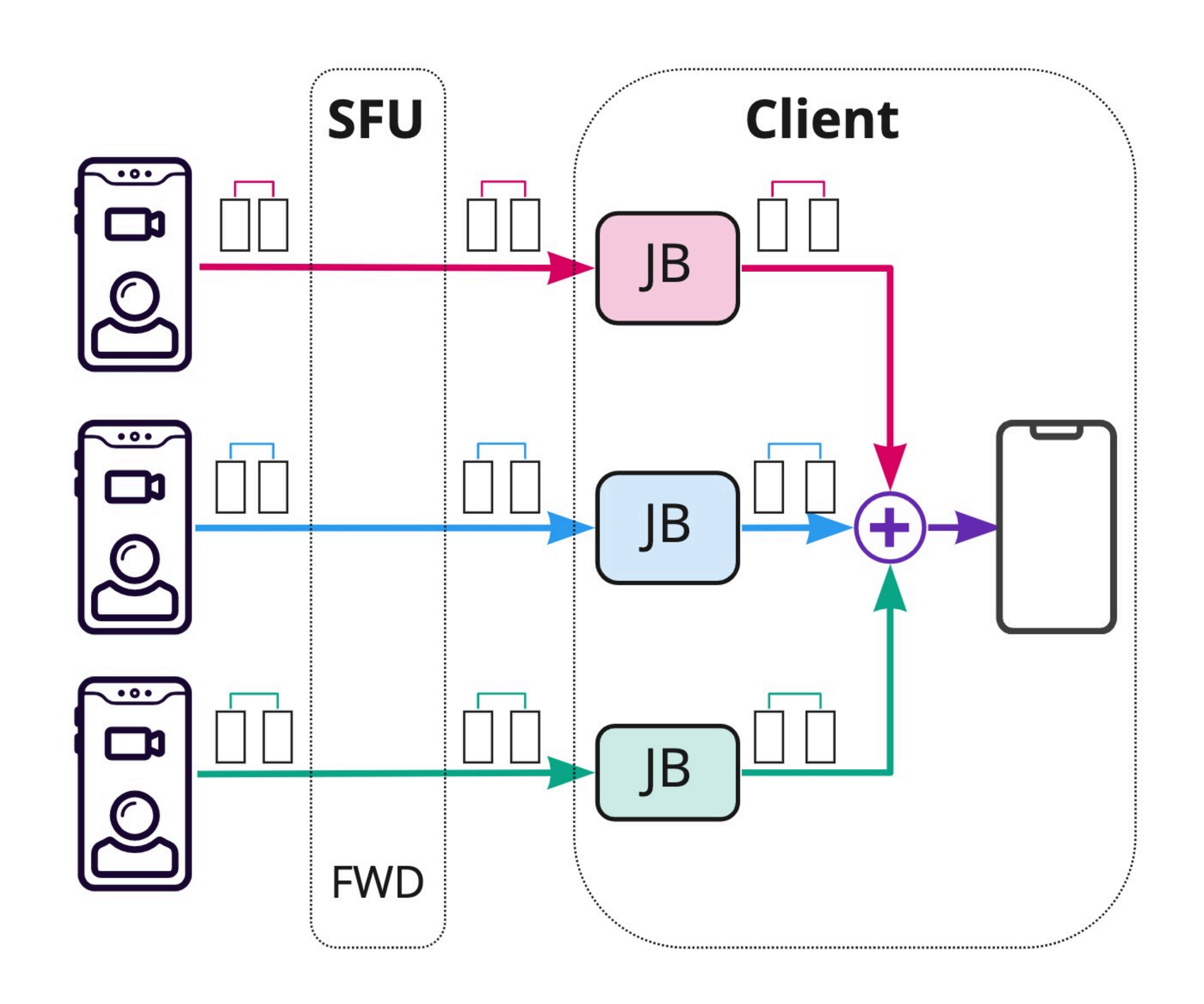




1. Выравнивает аудиопакеты по времени
2. Восстанавливает потери — PLC, FEC
3. Компенсирует задержку



SFU(forwarding) vs MCU(mix)



MCU

JB

JB

JB

JB

MIX

- много ресурсов на клиенте, 50+ не смиксуем
- 2 jitter buffer — выше задержка



Итого audio MIX

- На всех клиентах **один** audio поток
- У каждого voip инженера — свой **JB**)

0,09 CPU на одного участника:

- **decoding 0,04** CPU
- **mix** < **0,01** CPU
- **encoding 0,04** CPU

Задержка:

- **mix 0**
- **JB** зависит от клиента **p50 — 60 мс**



Audio tricks

Настройка кодирования под битрейт

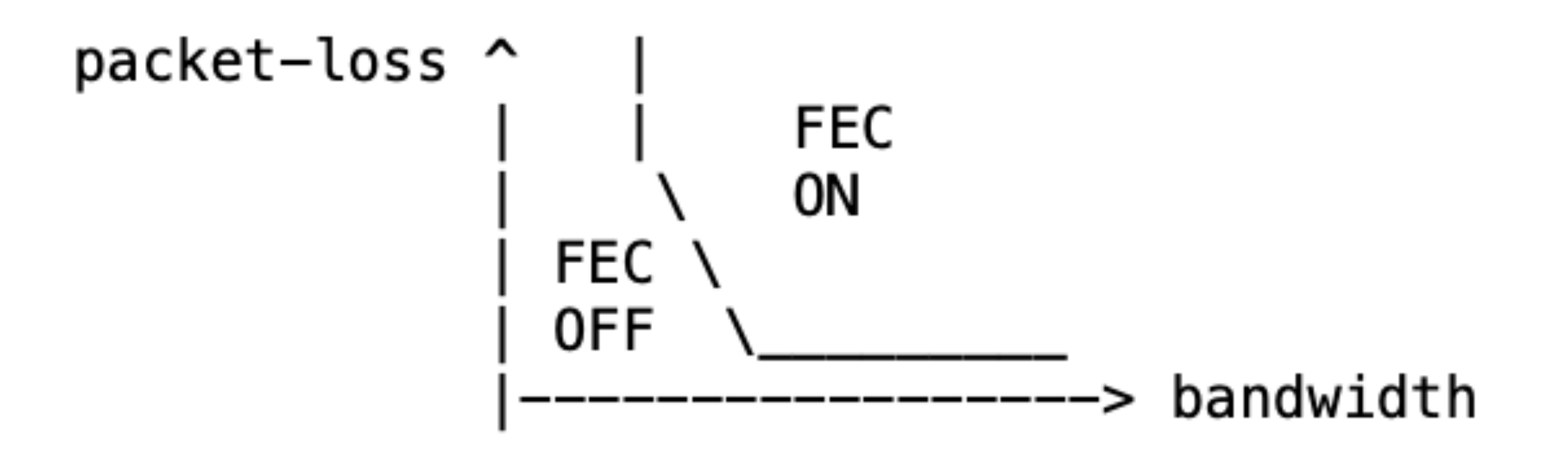


- WebRTC — **const** audio bitrate
- **AudioNetworkAdapter**
- Передаем **protobuf**-конфигурацию в **googAudioNetworkAdaptorConfig**
- https://github.com/jitsi/webrtc/blob/master/modules/audio_coding/audio_network_adaptor/config.proto









Адаптивный Forward Error Correction при больших потерях

```
     DTX   |   NO DTX

      mono |  stereo

-------------------> bandwidth
```

Не отправляем пакеты, когда участник молчит





Результаты mobile*

- Audio bitrate adaptation
- adaptive FEC (no NACK!)
- Packet size adaptation
- adaptive DTX | Mono

По мобильным клиентам:

- PL **–50%**
- RTT **–20%**
  - wifi: **220ms –> 170ms**
  - LTE: **255ms –> 205ms**
  - 3G: **470ms –> 360ms**



Практика:
audio pipeline

Audio pipeline



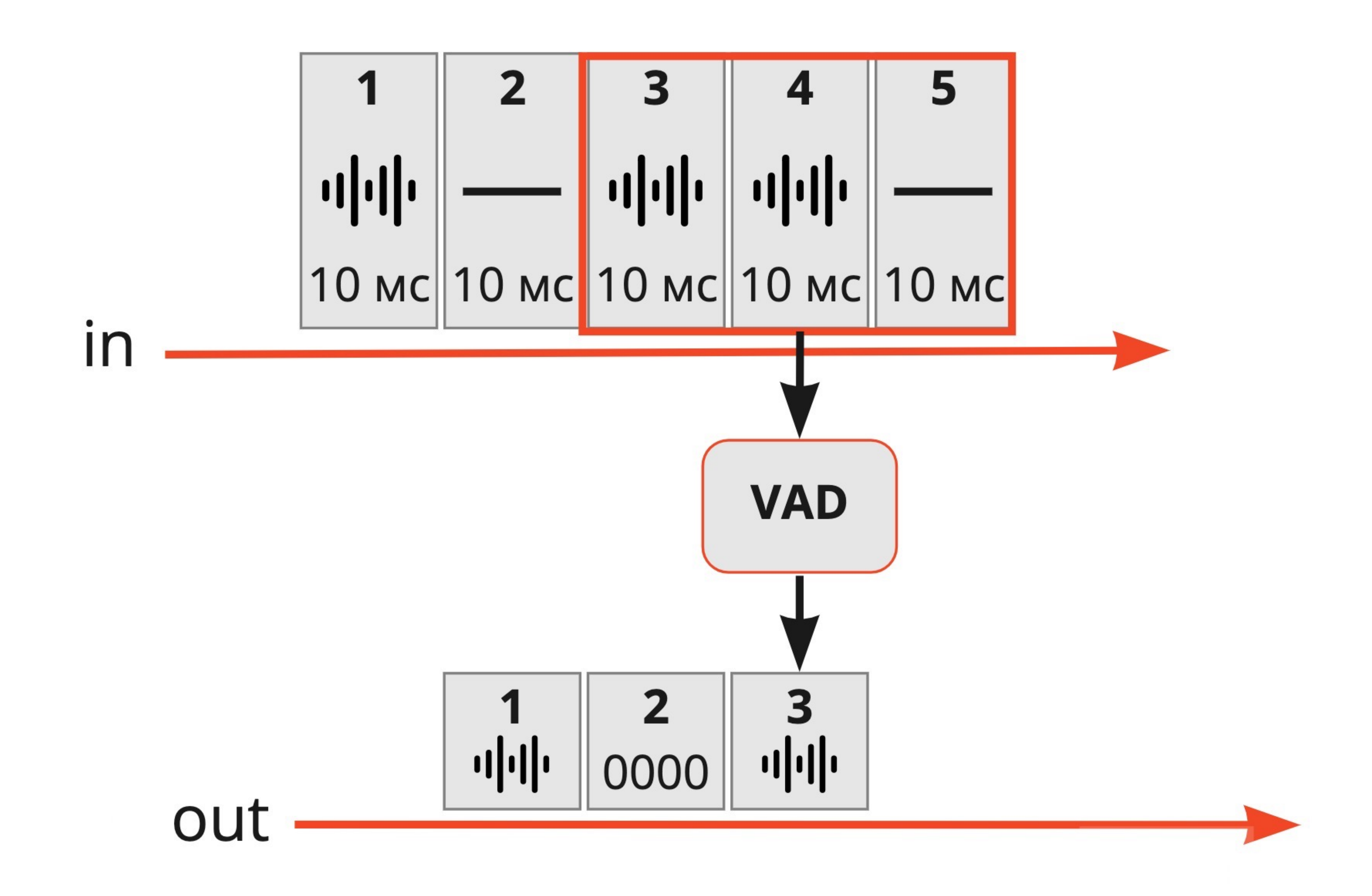


Зачем свой NS и VAD?

- WebRTC VAD на GMM
- WebRTC NS — heuristics algorithm
- Чем лучше NS и VAD, тем:
  - меньше передаем трафика,
  - меньше mix’им,
  - JB догоняет без искажений,
  - не мешает шум.



VAD на градиентном бустинге



На сложном тесте при
false_rejection = 1%

- **GMM**@WebRTC
  false_acceptance = **83%**
- **CatBoost**@VK
  false_acceptance = **55%**







Шумоподавление

Требования к технологии шумоподавления



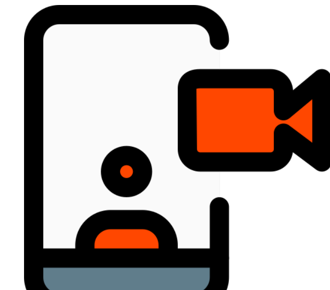
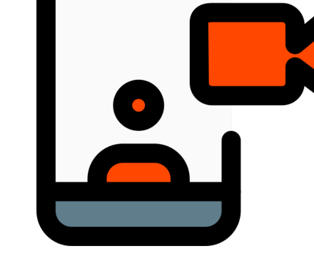
- RealTime
- Delay <= 20ms



- Качество шумоподавления на уровне SOTA
- ref krisp.ai



- min CPU
- min ROM



Замеры

Наша нейросеть в сравнении с другими популярными решениями

	PESQ	ROM
Исходный сигнал	**1,71**	—
webRTC	**1,8**	0
RNNoise	1,93	0,35 Мб
NSNet	2,05	11 Мб
DTLN (baseline)	2,41	3,9 Мб
DTLN (VK)	**2,54**	**5,3 Мб**

PESQ-метрика ≈ экспертной оценке по шкале от 1 (плохо) до 5 (отлично).

ROM влияет на размер приложения.



Шумодав Итого

- Своя реализация **DTLN**-подобной архитектуры на **C++**, которая работает быстрее: **30 мс** за **~0,2 мс**
- Используем окно **30 мс** и перекрытие **20 мс** для бесшовной интеграции с WebRTC и **Opus**
- Модель **5,3 Мбайт**
- Отстаёт по качеству от лучших моделей не более, чем на **10%**
- **On-demand** закачка на телефон

В течение месяца будет статья на https://habr.com/ru/company/vk/



эффект VAD + NS

−30%
аудиотрафика

−60%
ускорений с искажениями

+40%
качества по PESQ





- Клиентский CPU
- Browser



Клиентский шумодав, jitter и CPU

01:00 02:00 03:00 04:00 05:00 06:00 07:00 08:00 09:00 10:00 11:00 12:00 13:00 14:00 15:00 16:00 17:00 18:00 19:00 20:00 21:00 22:00 23:00 24:00

Запуск шумодава

Искажения для компенсации задержки на сервере



p75 audio jitter

from client

to client

Решение

- native audio pipeline
- back pressure CPU (батарея)



всегда смотрите метрики



Какие catBoost'ы и нейросети в браузере?

client

Acoustic Echo Cancellation

Audio encoder (Opus)

server

Audio decoder

Automatic Gain Control

VK Voice Activity Detection

VK neural network noise suppression

- **0,1 CPU** на говорящего пользователя после VAD
- WASM





Audio pipeline: результаты

0,09 CPU
на пользователя

+40%
качества по PESQ

–30%
меньше аудиотрафика



Какая у нас задержка на сервере



p50
60 мс

p99
500 мс

passthrough latency

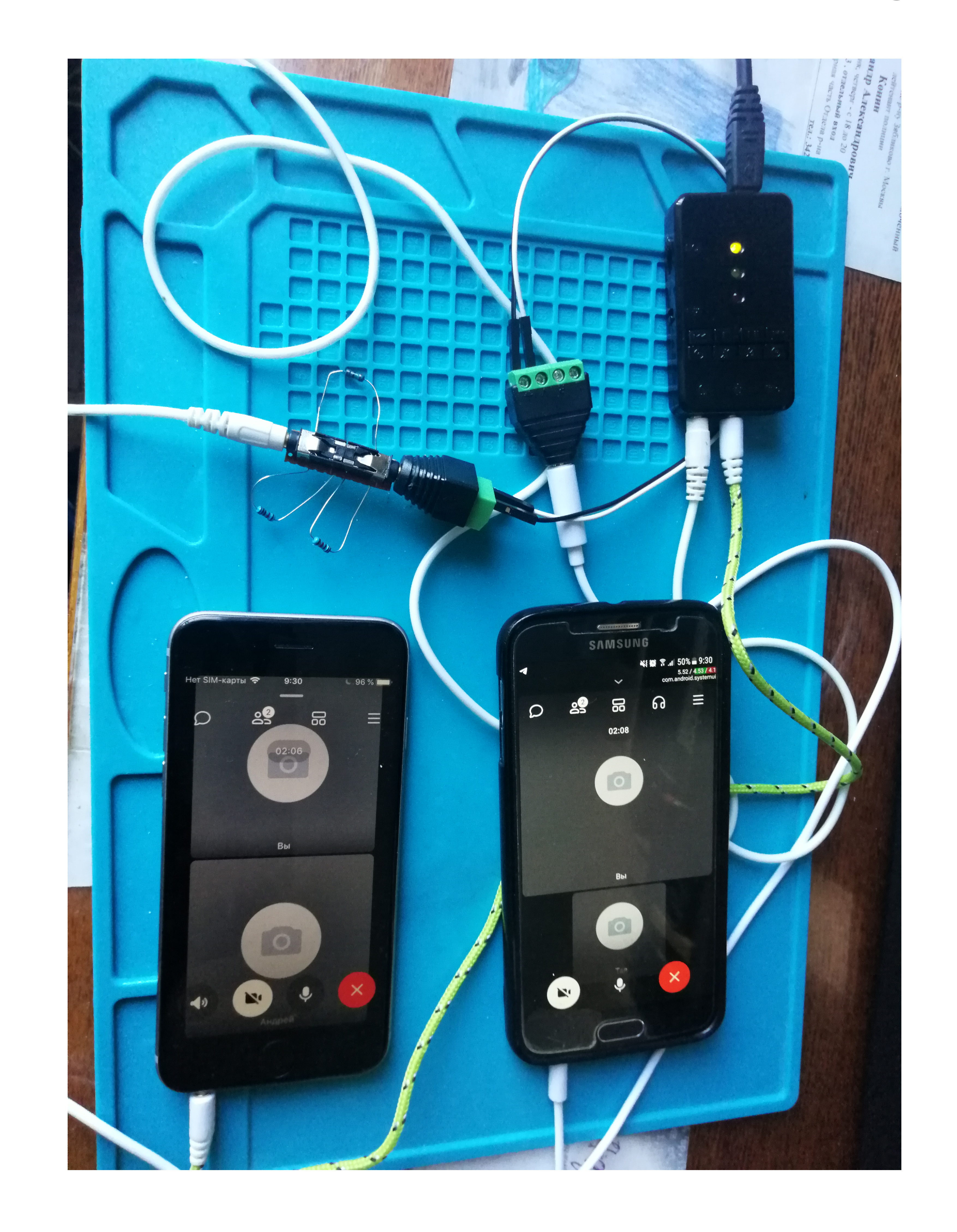
20 мс

VAD + NS latency



Где мы находимся относительно конкурентов?

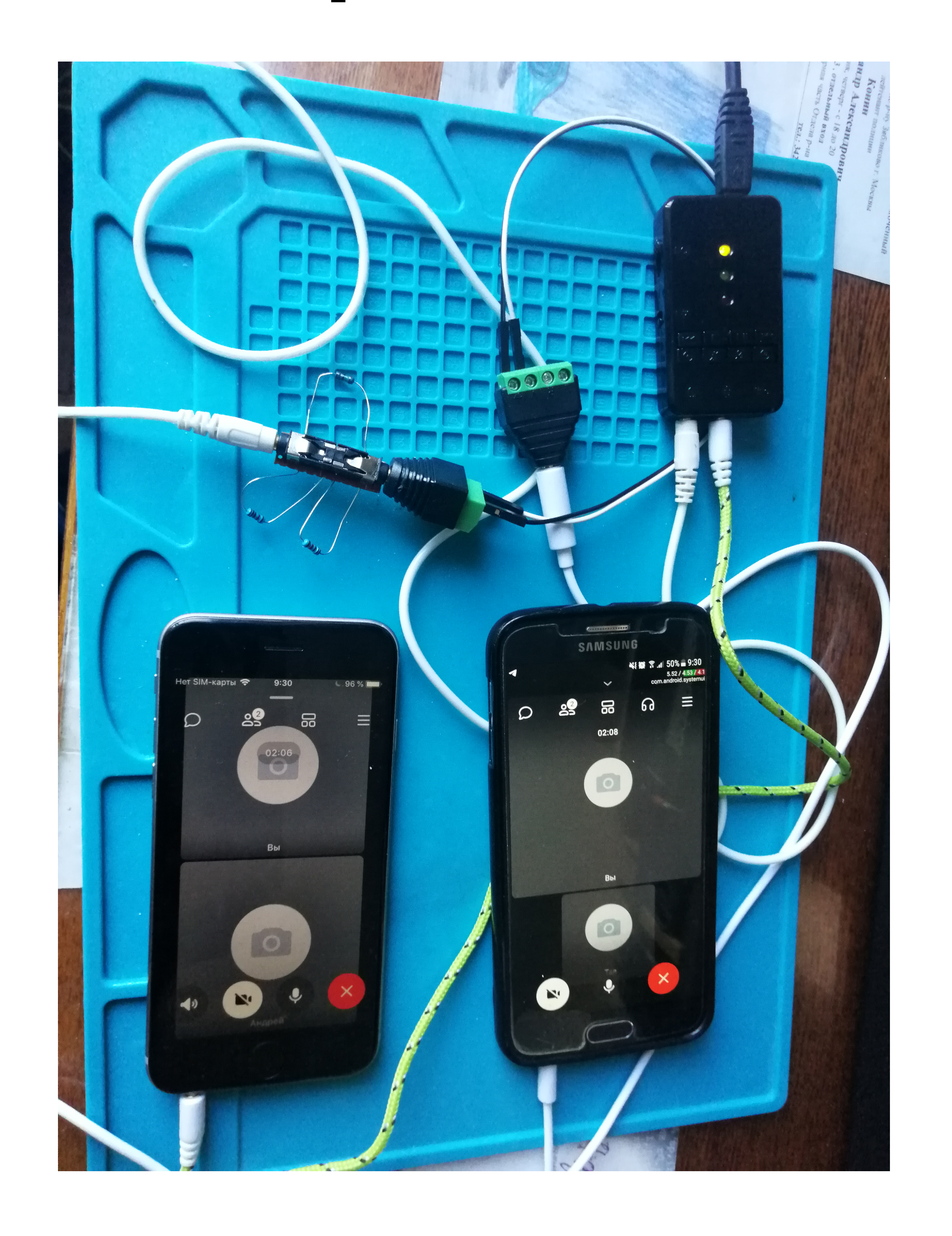
Как померить аудиозадержку?

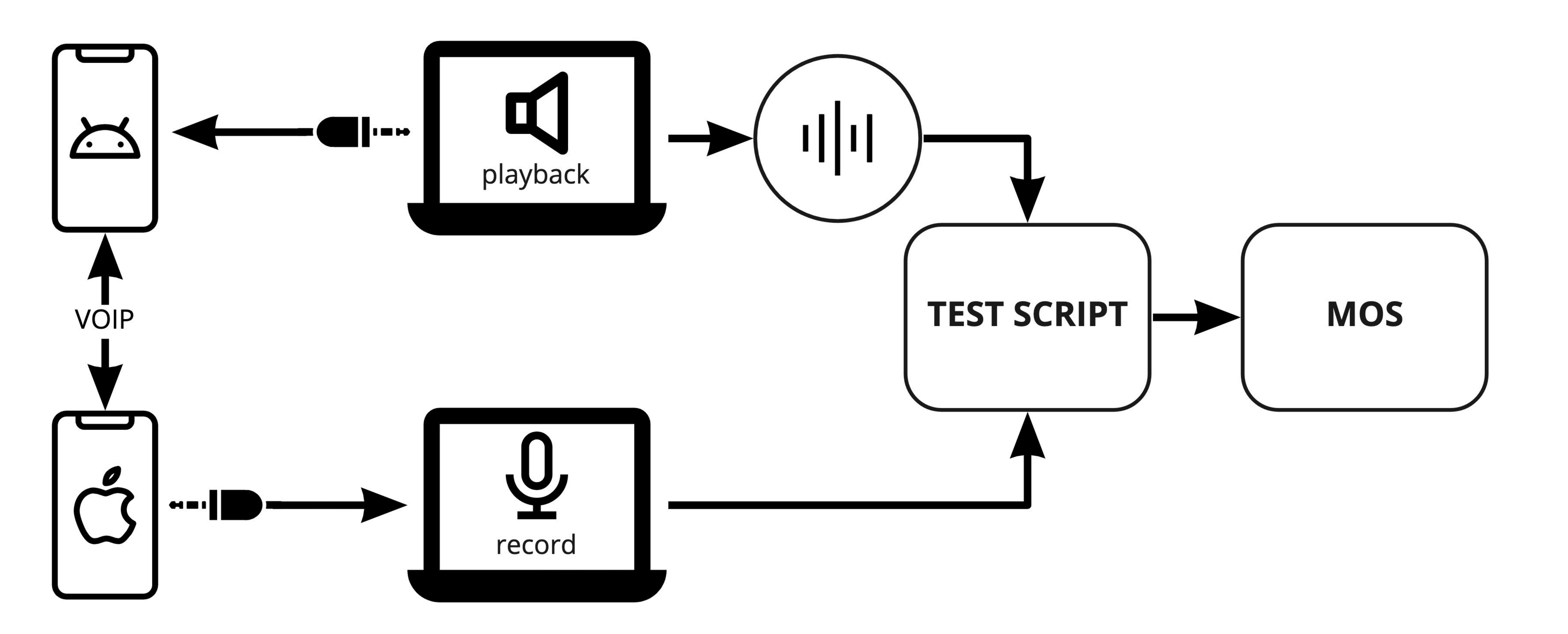






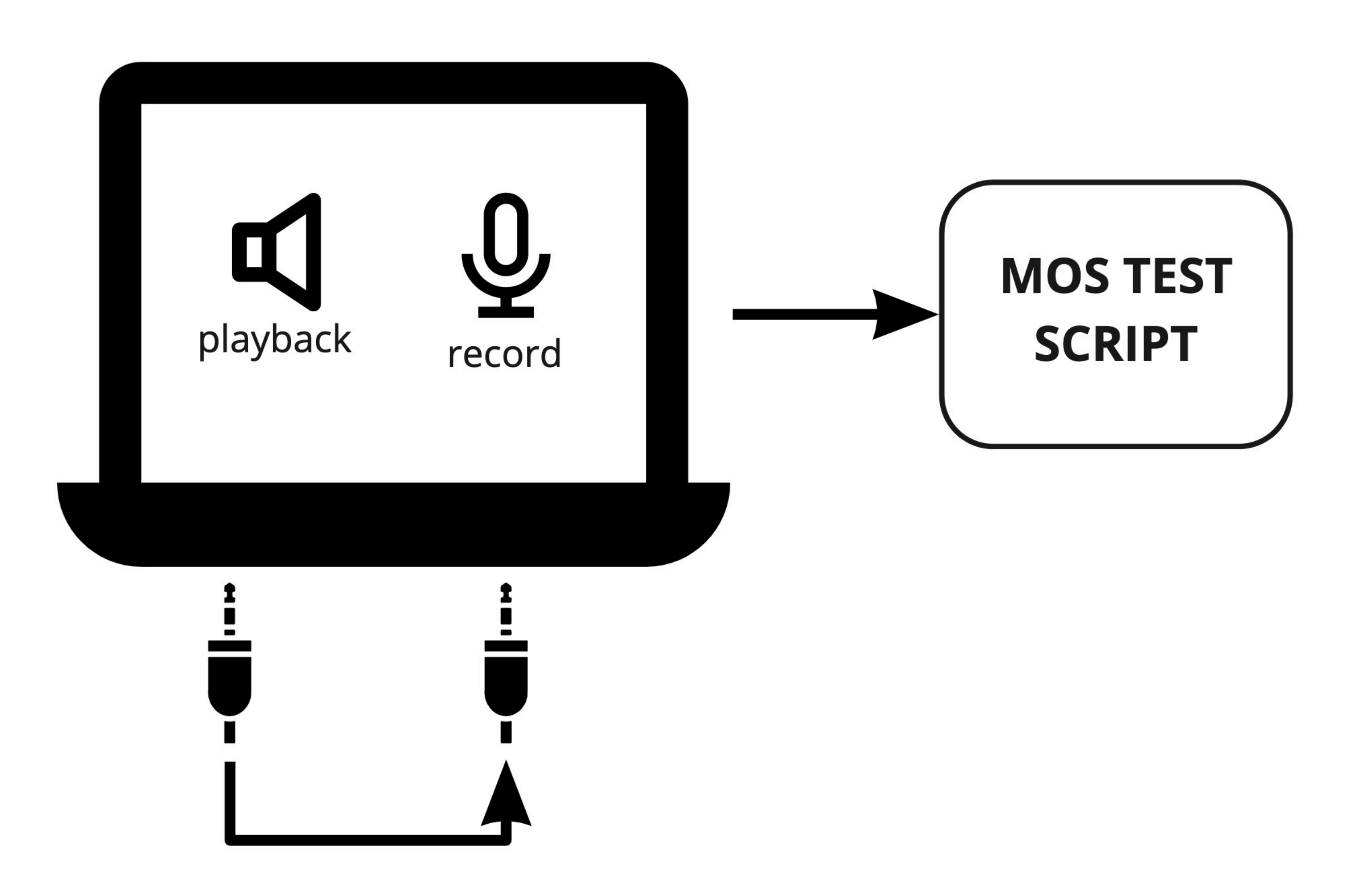
Измеряем mobile latency







Как измерить desktop latency



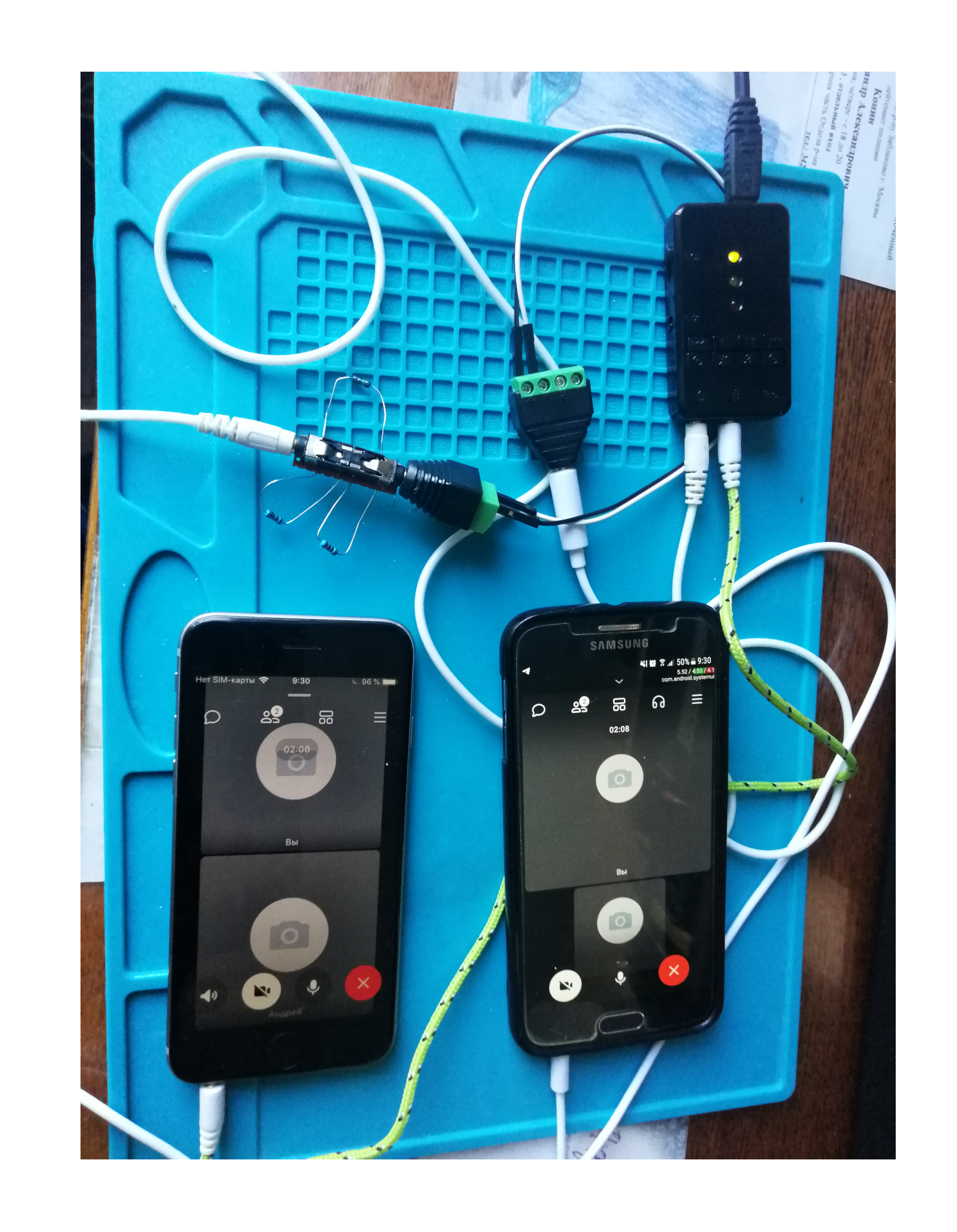
Virtual Audio Cable (VAC)
https://vac.muzychenko.net/en/index.htm

miro



Сравниваем с конкурентами

	Delay minus loopback (ms)	MOS	# of delay jumps
loopback, no VOIP	355	2,58	1
VK	**212**	**2,93**	**17,3**
Google Meet	246	2,72	18,5
Viber	255	2,21	15,8
WhatsUp	338	2,39	11,1
Skype	385	2,2	17,7
ZOOM	428	2,12	19,1
Discord	496	3,05	16,5





Теория

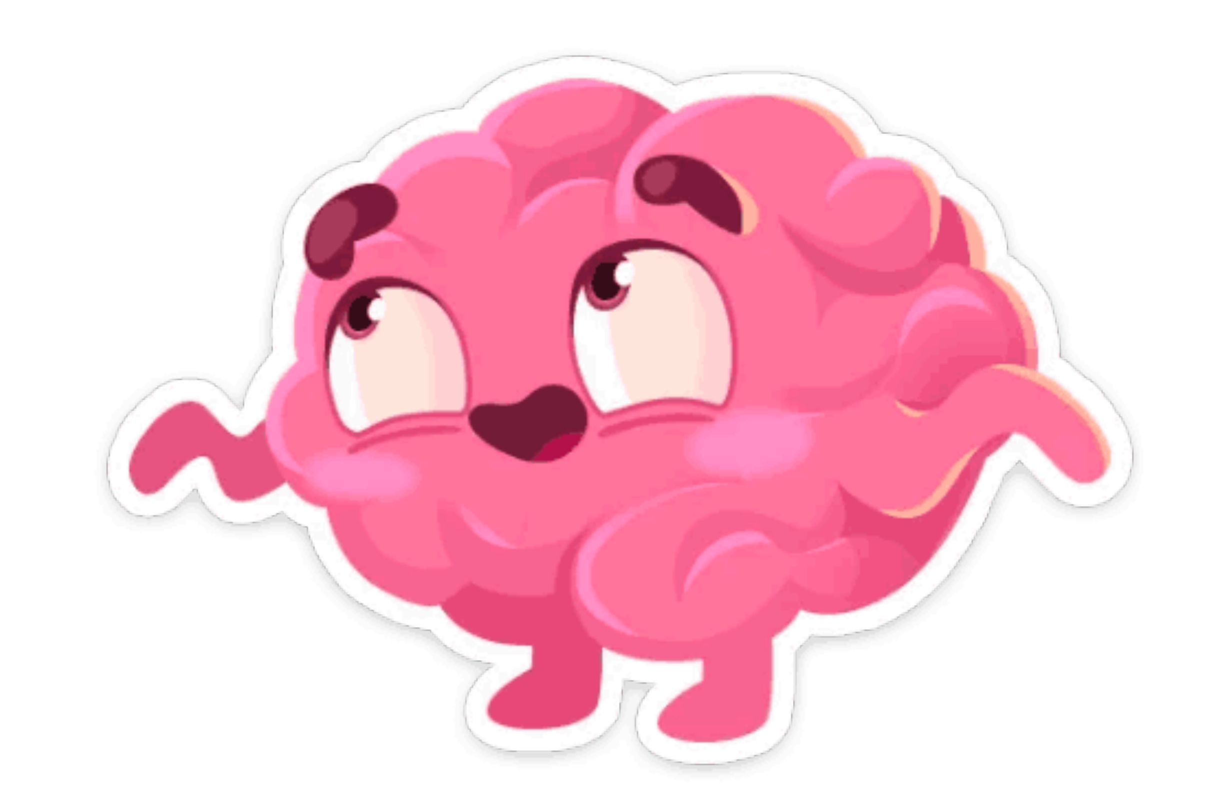
	MIX	FWD
latency	max	avg
max participants	∞	~ 50
uses in	**Google Meet, VK**	**Zoom**

chrome://webrtc-internals

https://zoom.us/docs/doc/Zoom_Global_Infrastructure.pdf



Практика — групповой звонок



	Value
Google meet (web)	390
VK (web - native client)	431
VK (web)	454
Zoom (web - native client)	609
Zoom (web)	685

Axis: 0, 200, 400, 600, 800

Теория	Google Meet, VK	Zoom
latency	max	avg

*сеть до сервера выровнена



Audio Trick

Теория	**Google Meet, VK**	**Zoom**
max participants	∞	~ 50

ВСЕ СРАЗУ НЕ ГОВОРЯТ

+ чистый SFU — низкий latency

– чем больше участников говорит и шумит, тем выше Bitrate (нужны хорошие NS, VAD)



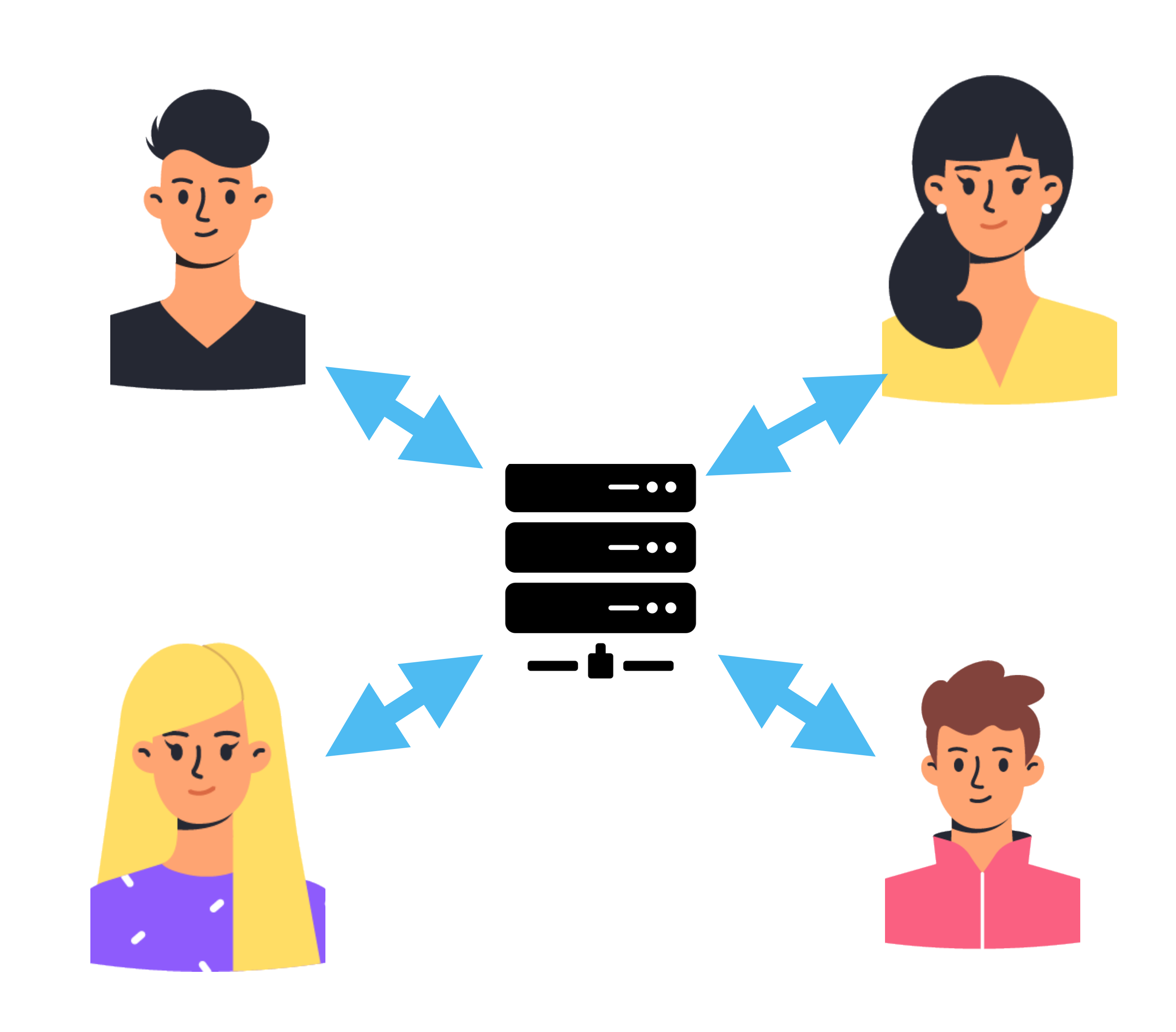
Выводы

- **MCU/Mix** дольше **не всегда**
- Не всё решает **топология**
- Мы ~~выше~~ на уровне рынка по **latency**, **# of delay jumps** и **MOS**
- Экономим сеть и **CPU** пользователей



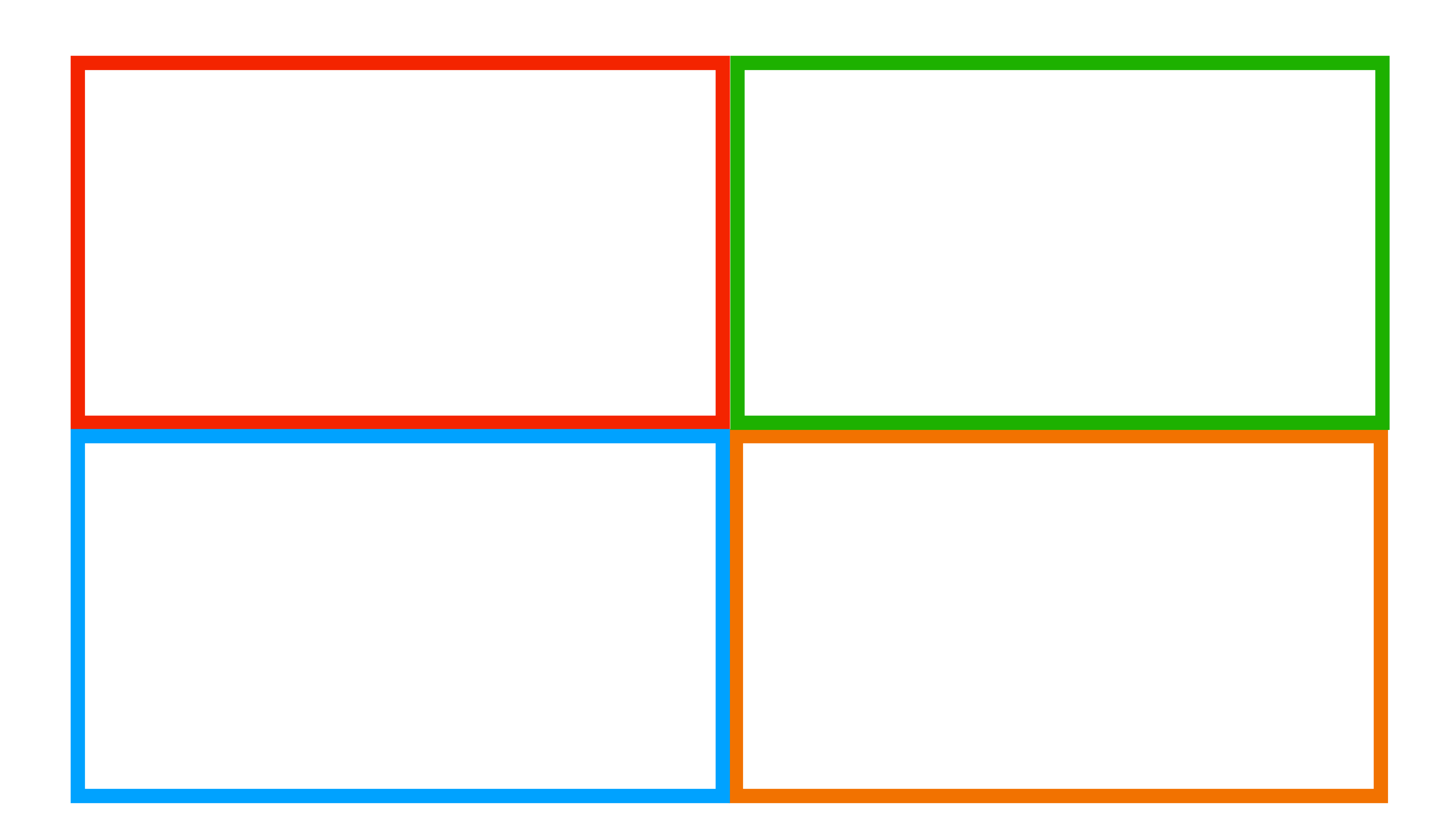
Видео

Video MCU — mix



MCU — MIX

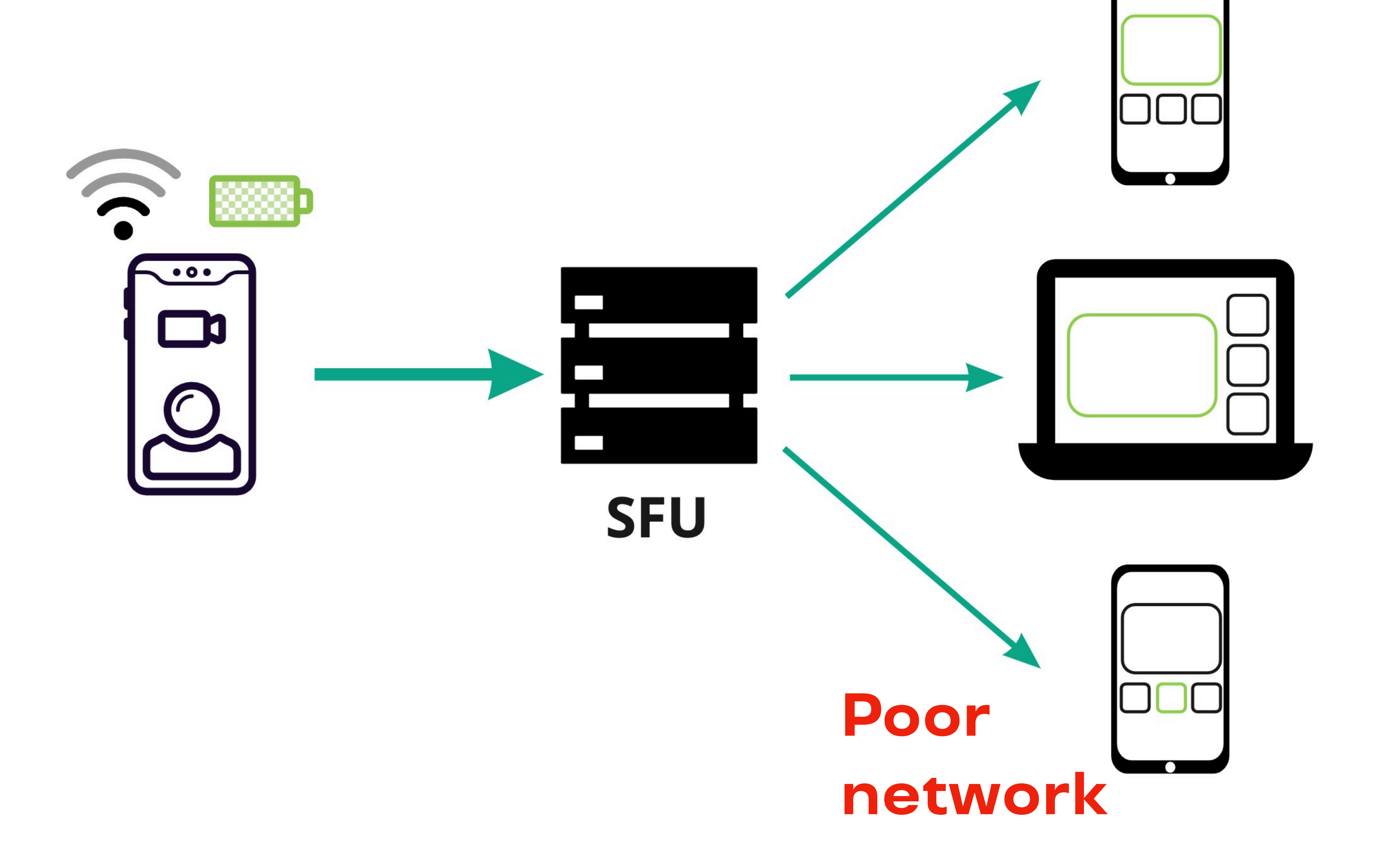
Multipoint
Conferencing Unit



ОЧЕНЬ ДОРОГО



SFU





- FIR storm
- Bandwidth adaptation



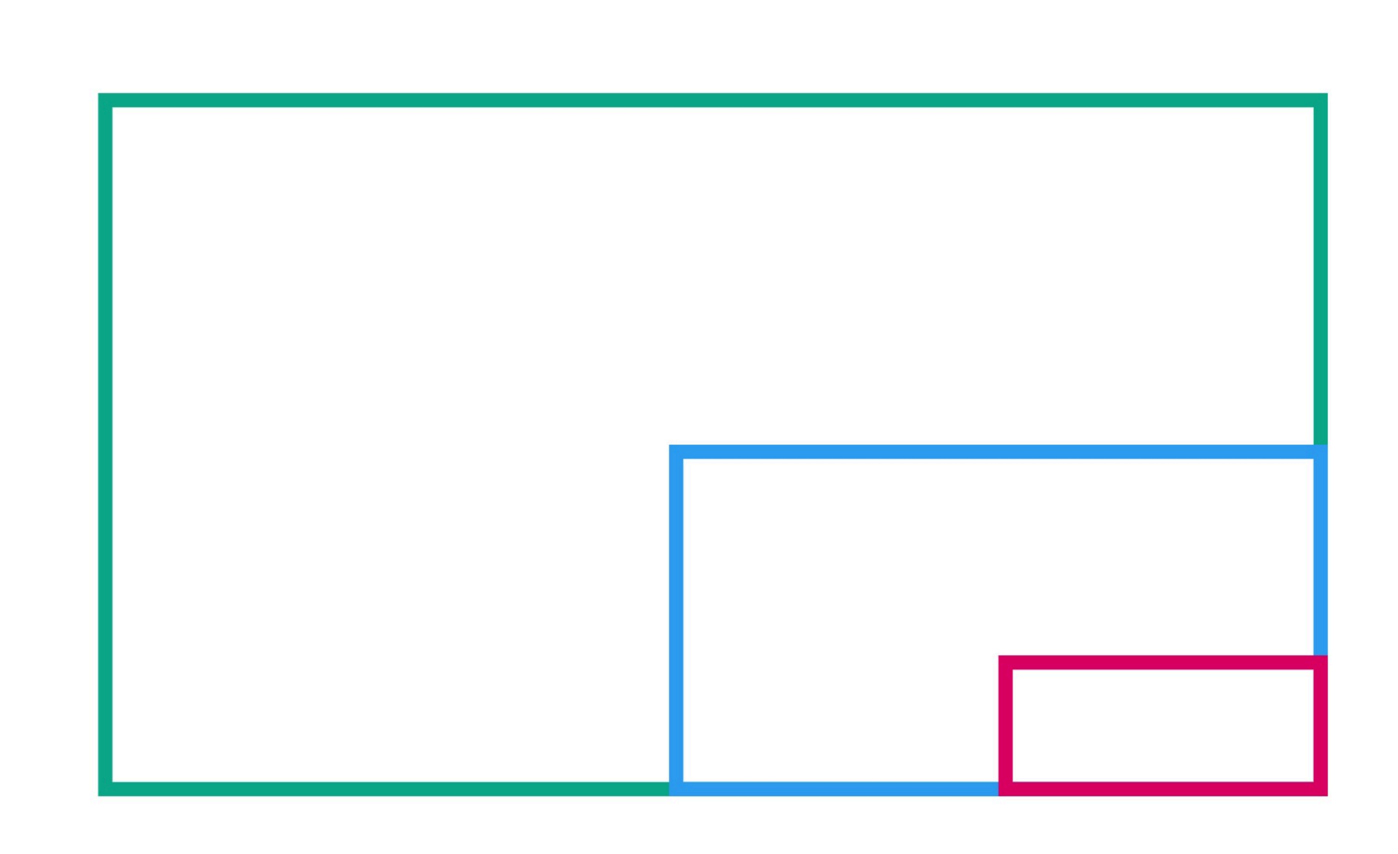
Simulcast транскодирование видео

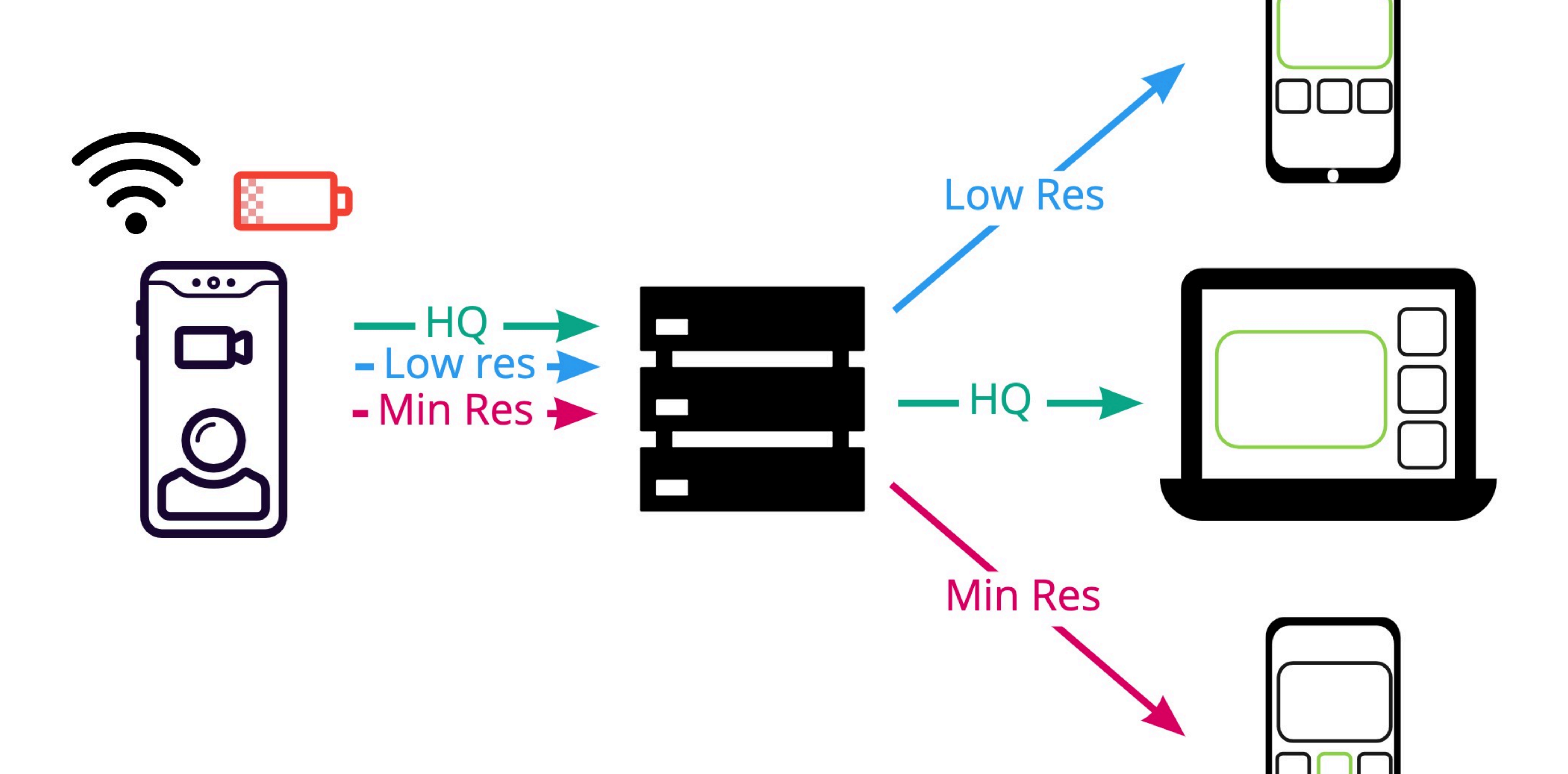
HQ
Low res
Min Res
Low Res
HQ
Min Res

- Bandwidth adaptation +
- FIR Storm +/—
- Много клиентских ресурсов —



SVC





- Bandwidth adaptation +
- FIR Storm –
- Меньше CPU/Traffic чем Simulcast
- Не поддерживается в браузере



Серверное транскодирование видео

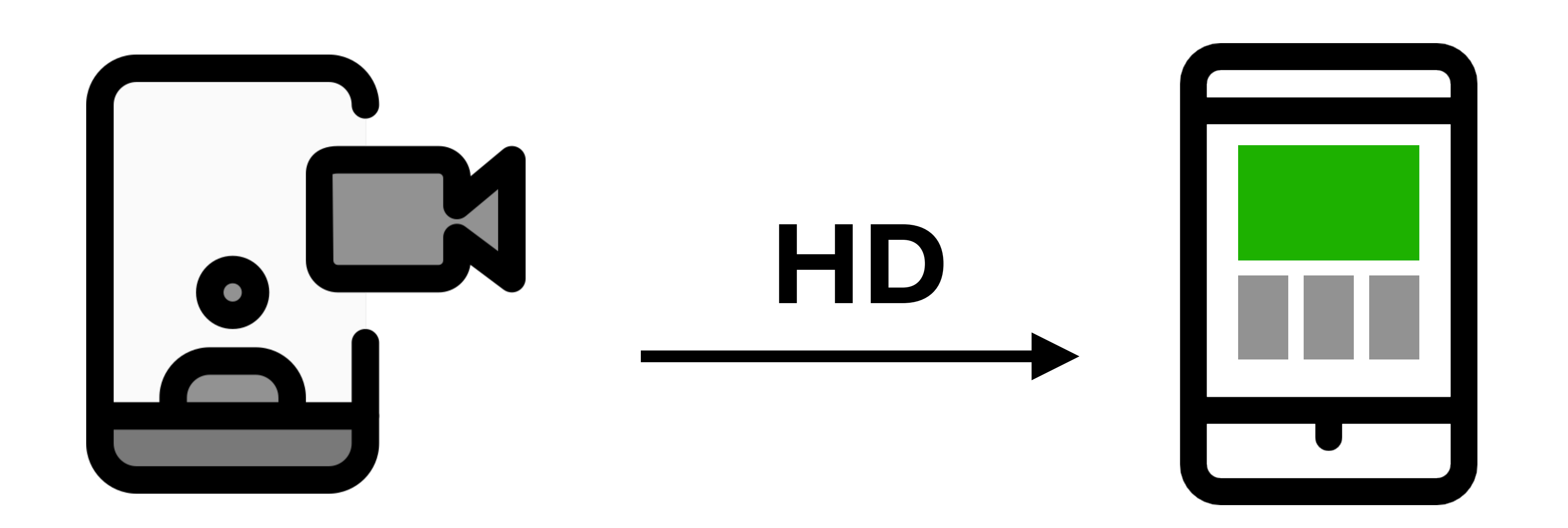
HQ
Low Res
HQ
Min Res

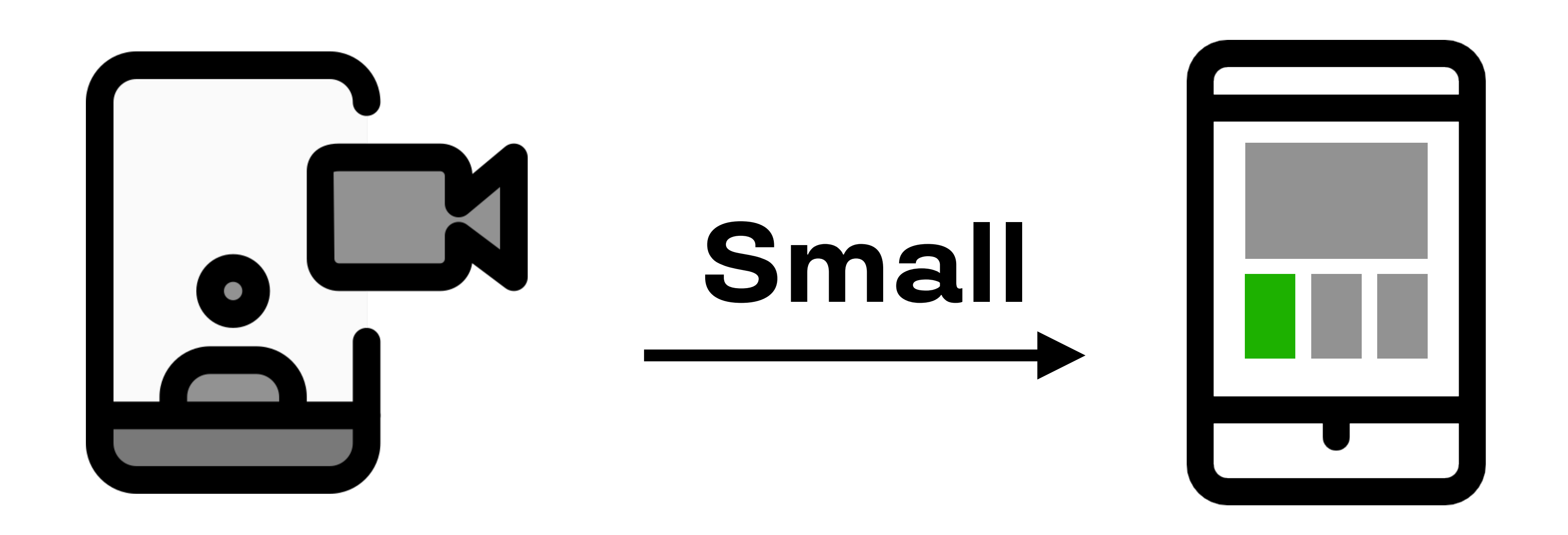
- Bandwidth adaptation +
- FIR Storm + (генерируем на сервере)
- дорого –

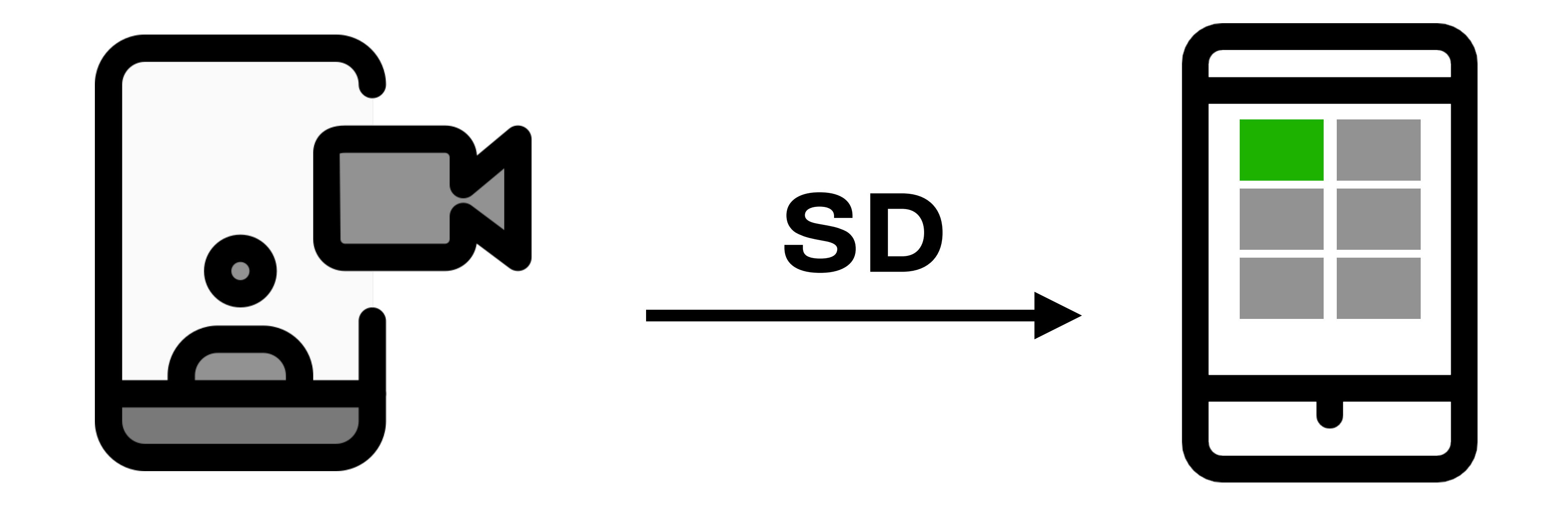


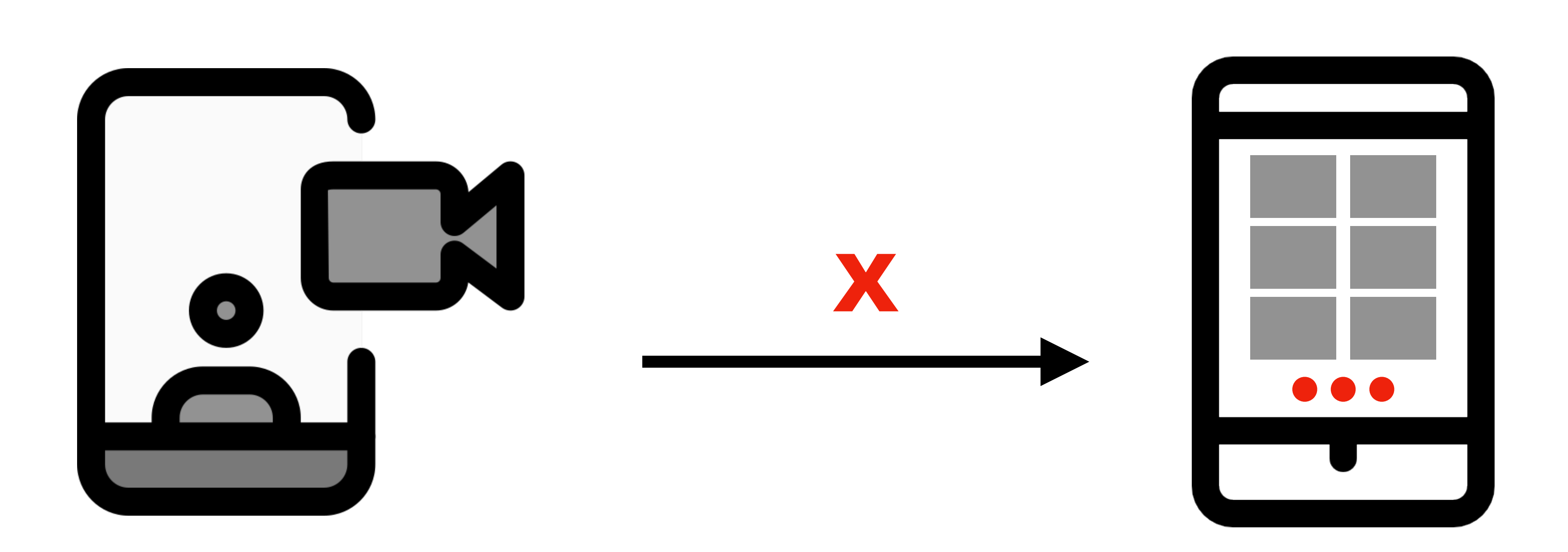
Quality on-demand

Качество по запросу



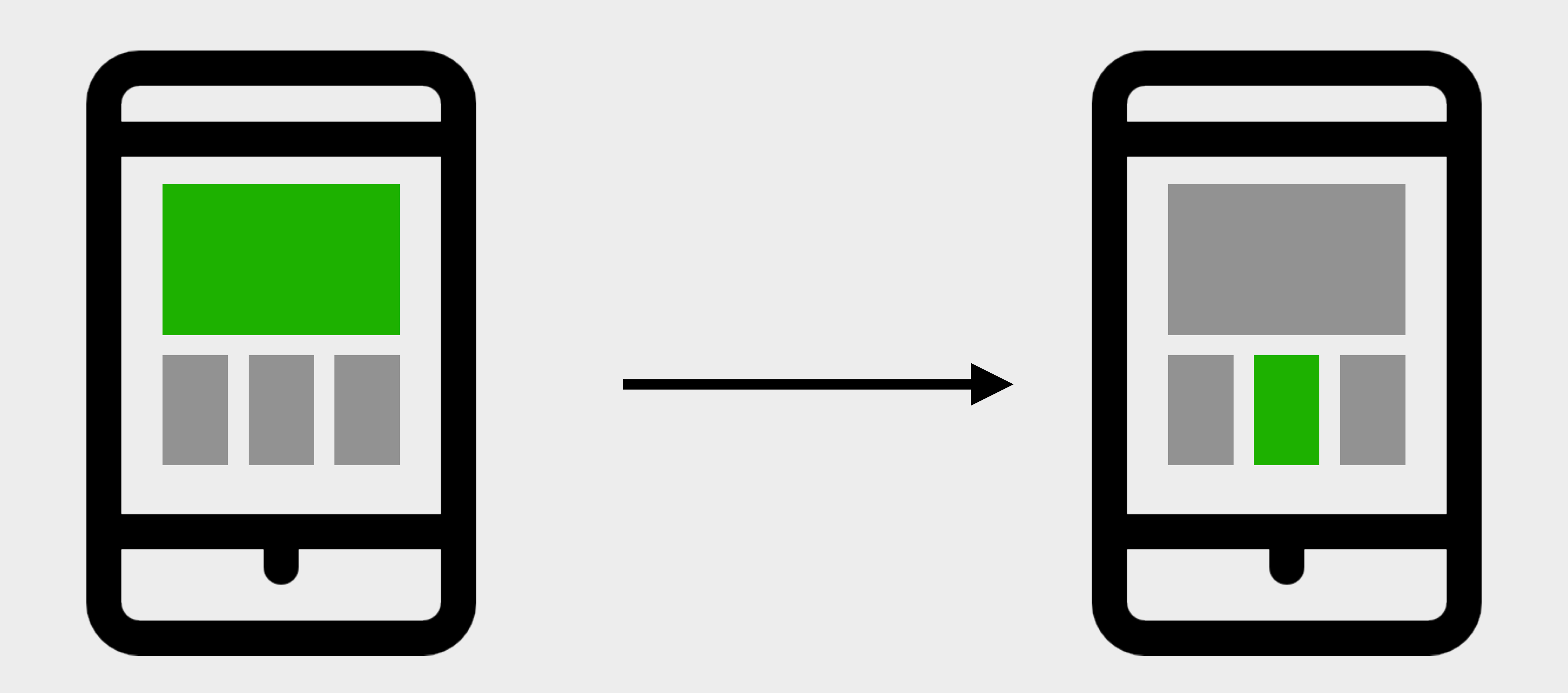


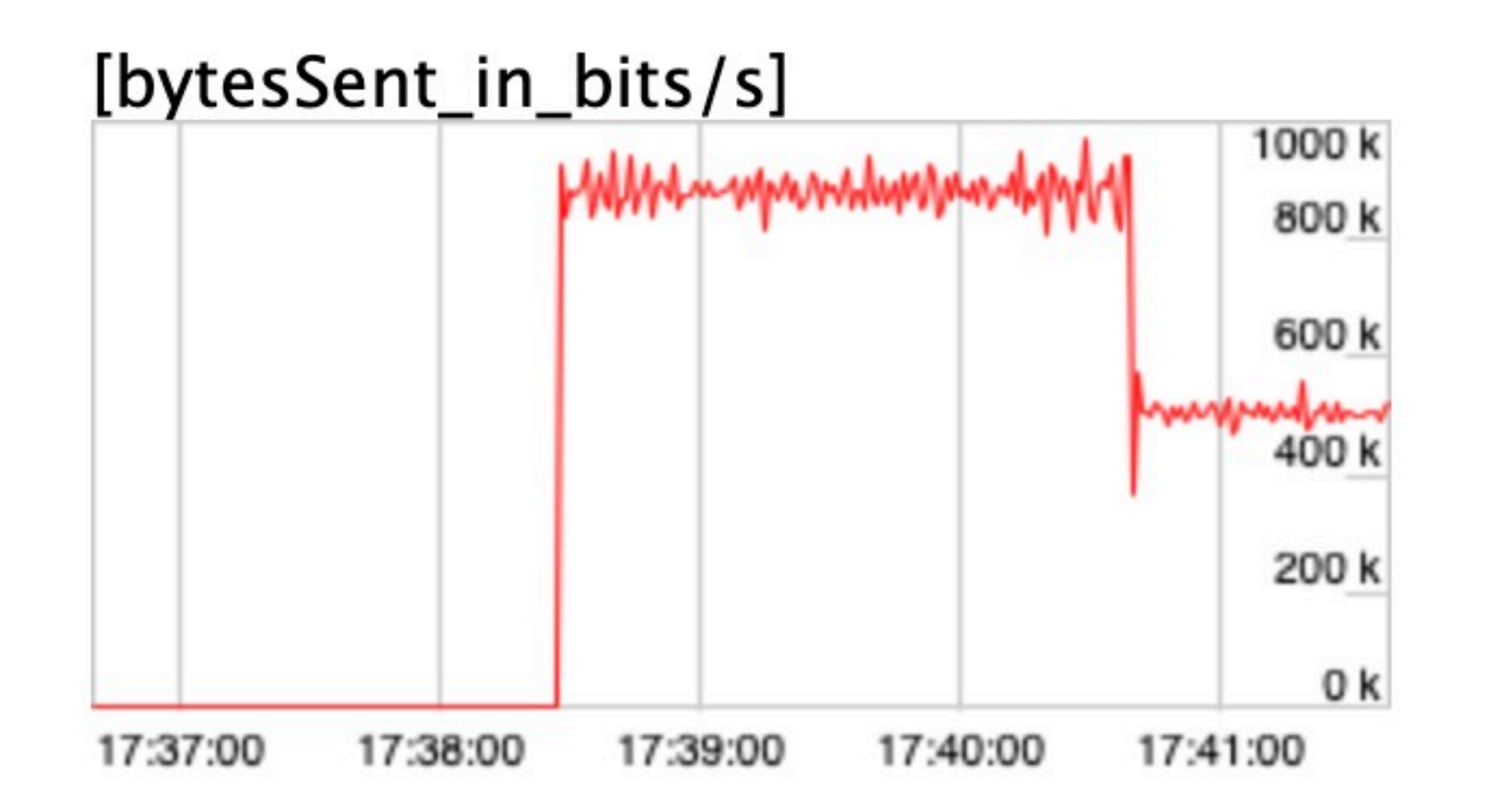


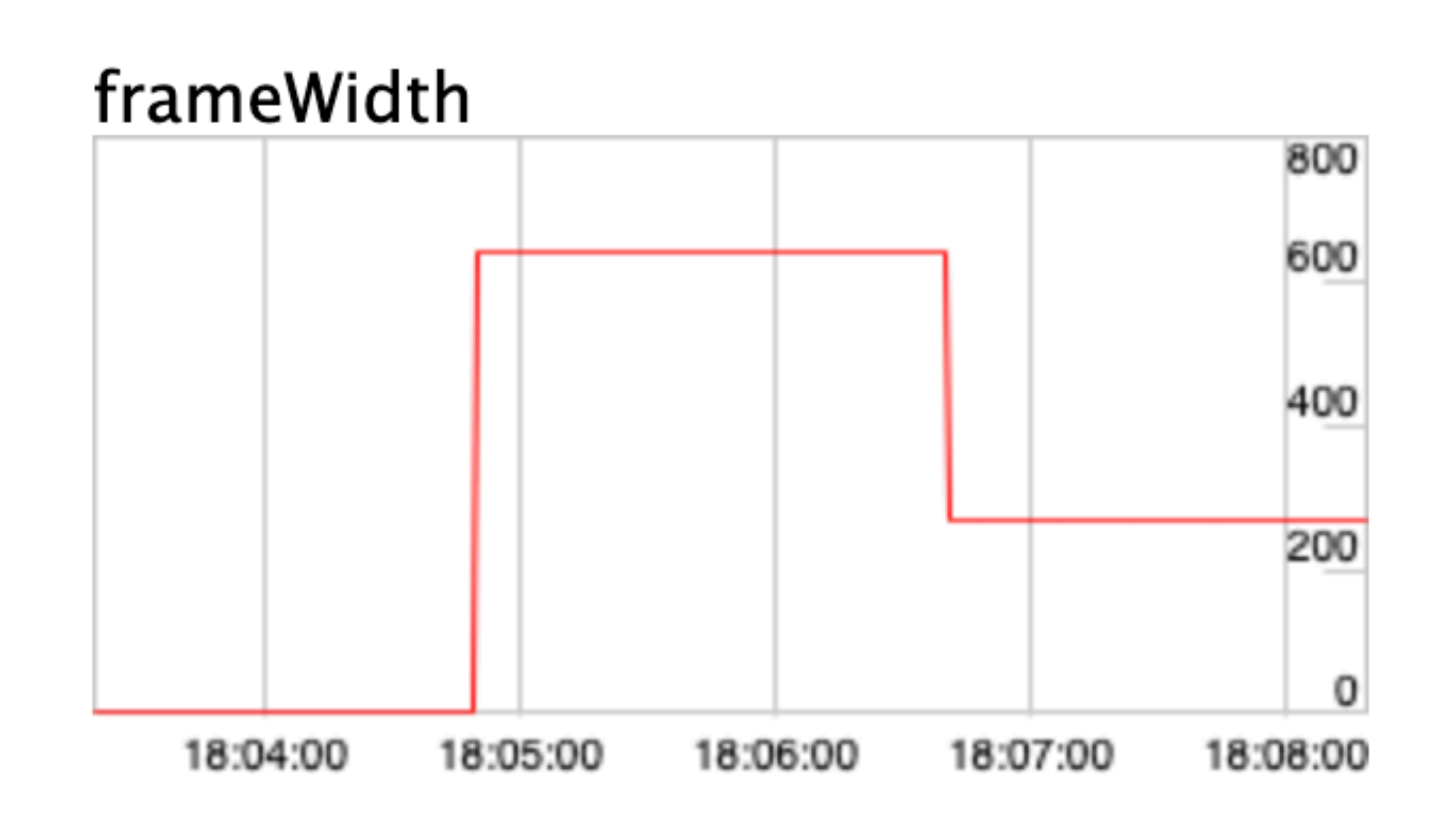


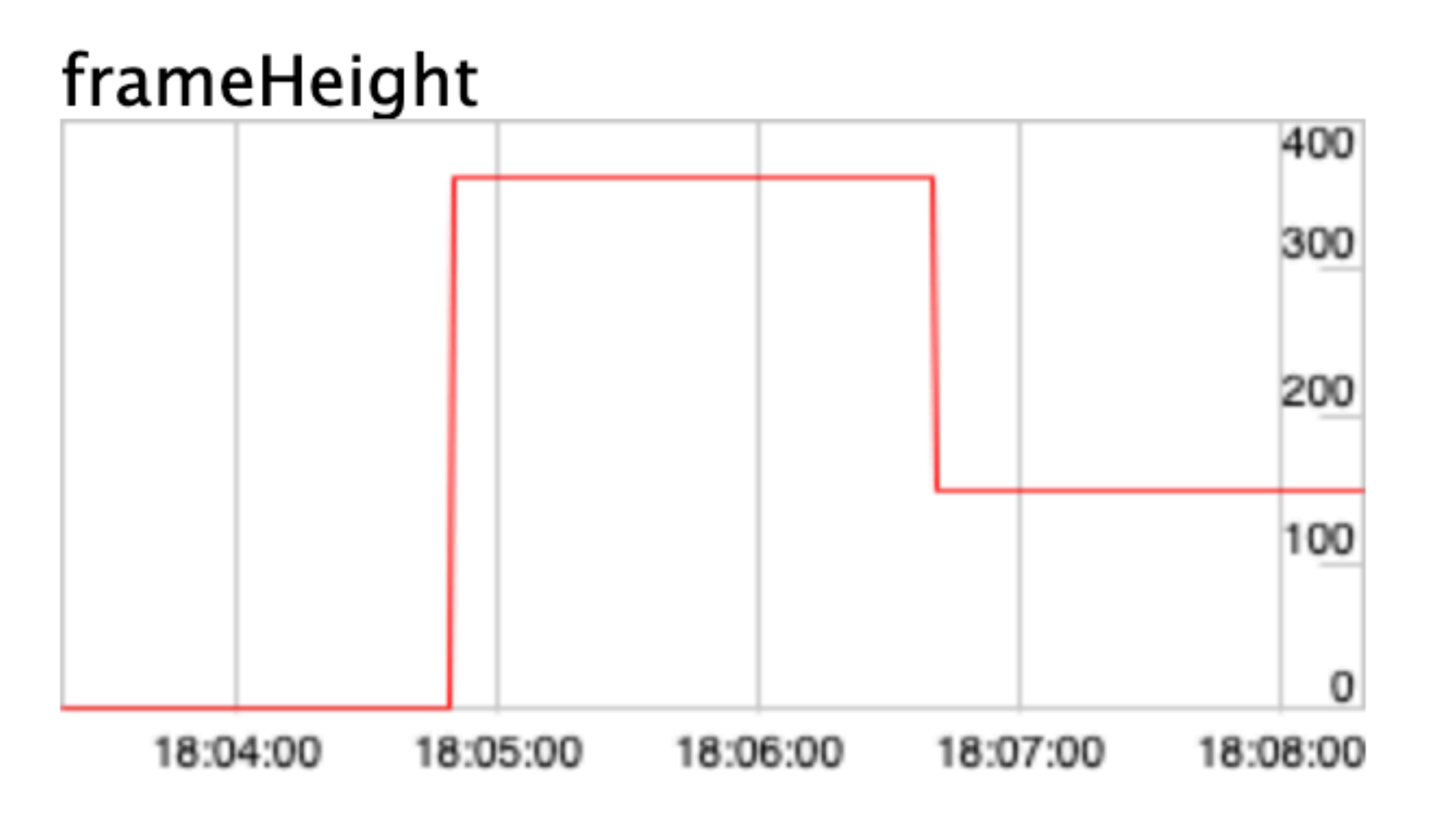


Переменное качество исходящего видео









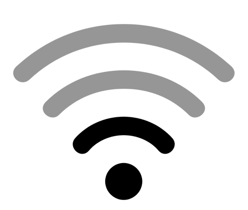
webrtc-internals





Кодирование видео на клиенте по запросу

Экономим:



62% client→server traffic.



21% client CPU.



12% server CPU.

Это в самом неоптимальном случае на 3 участника!



Серверное транскодирование по запросу



Low Res

c1

Min Res

Min Res

c2

Min Res

cN

Экономим:



40% не нарезается



Simulcast vs SVC vs Quality on-demand (QOD)

Simulcast vs SVC vs Quality on-demand (QOD)

	SFU	Simulcast	SVC	QOD
input traffic	3 Мбит/сек	1 Мбит/сек	1 Мбит/сек	1 Мбит/сек
output traffic	1 Мбит/сек	1,5 Мбит/сек	1,2 Мбит/сек	1 Мбит/сек
client CPU	100 %	200 %	150 %	100 %
server CPU	0	1 %	2 %	34 %



Выбор кодека

Настройки кодека — это компромисс



Качество и нагрузка на устройство

H.264 — быстрый, аппаратно-ускоренный кодек на iOS и Android

0 500 1000 1500 2000

Bitrate, kbps

VP9 — более новый, но пока не ускоренный аппаратно кодек

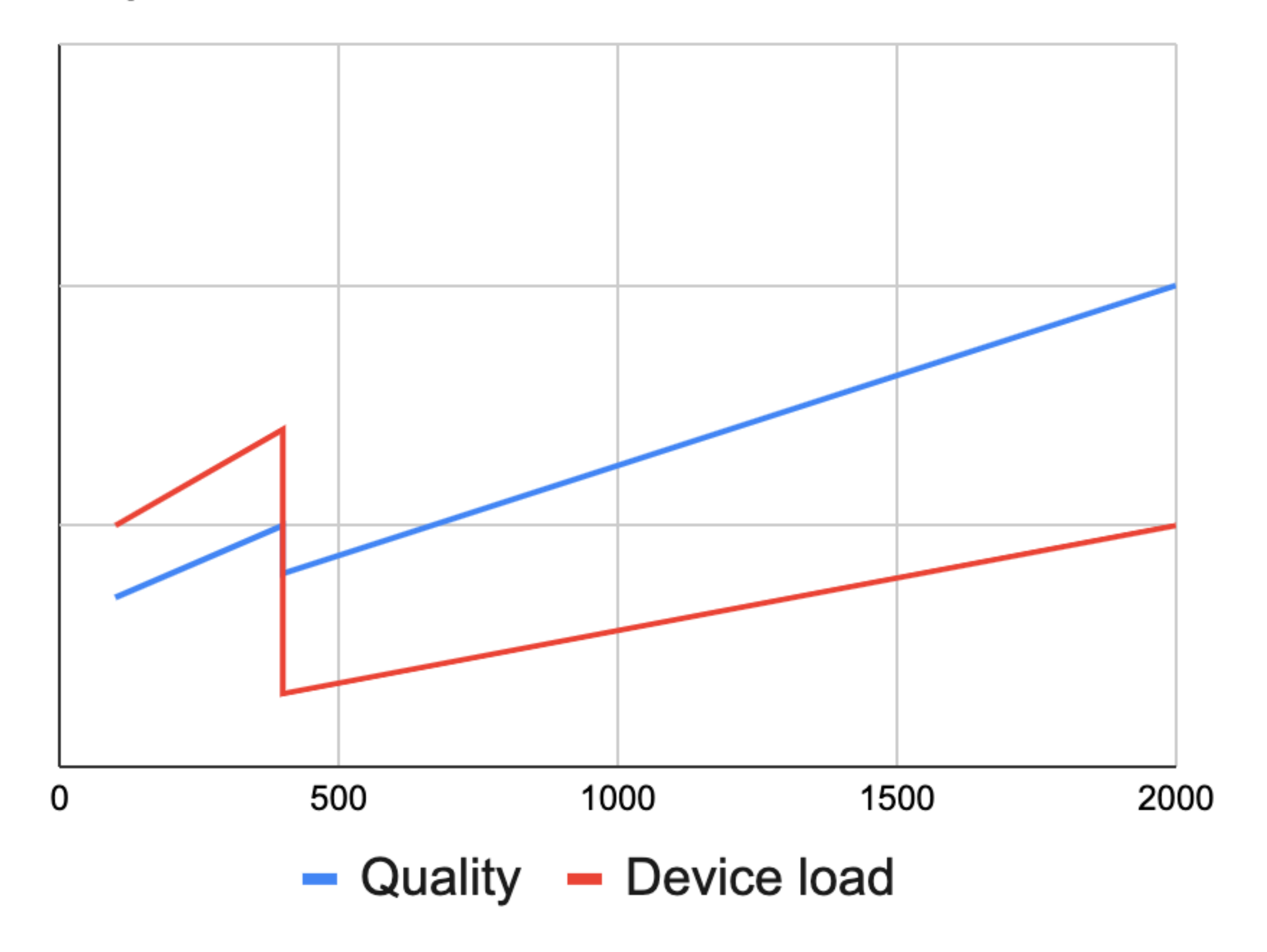
0 500 1000 1500 2000

Bitrate, kbps





Переключаем кодеки по битрейту



Если битрейт выше порога — H.264, если ниже — VP9:

- не греем телефон на высоких битрейтах,
- вытягиваем качество на низких.



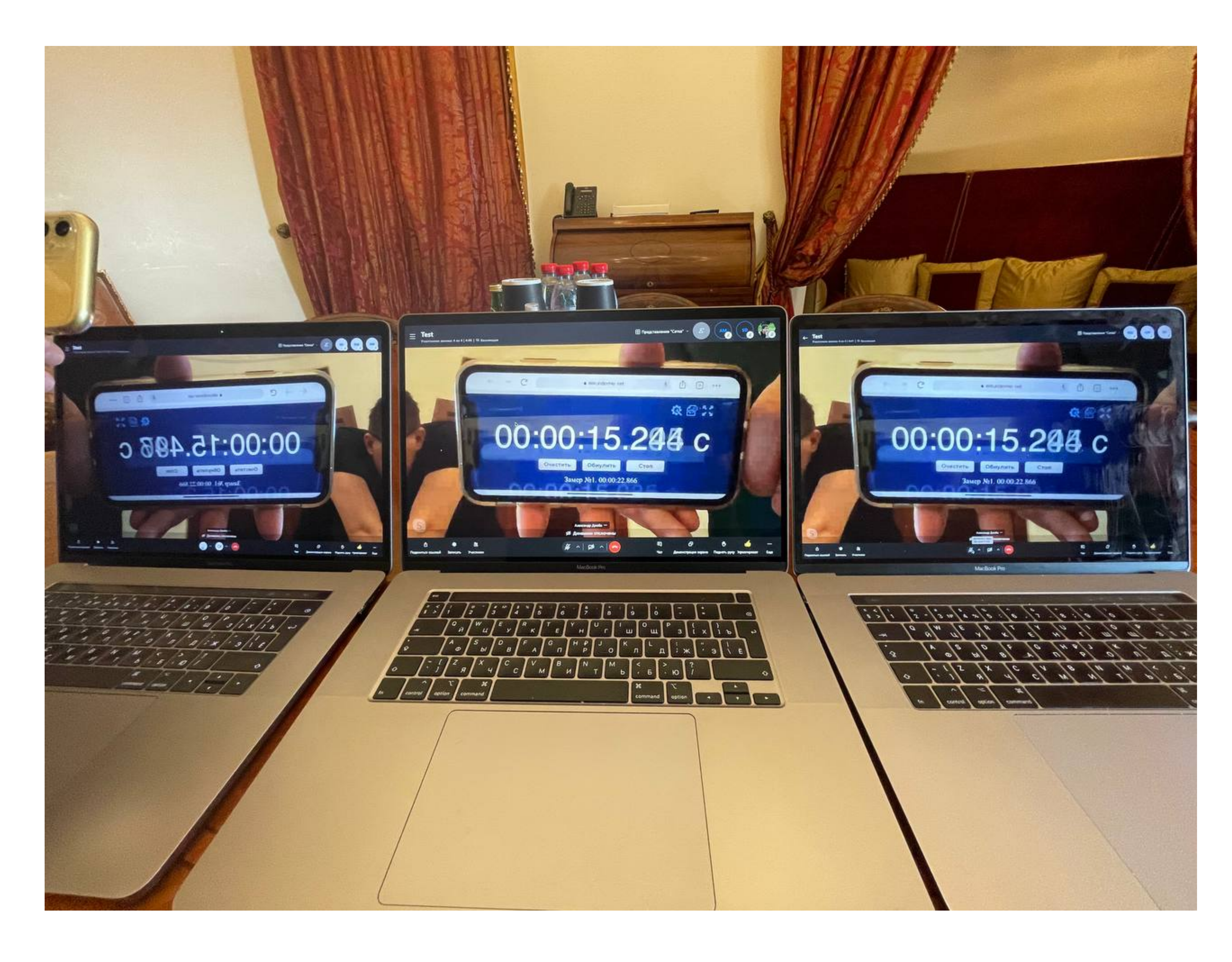
Video transform on demand: результаты

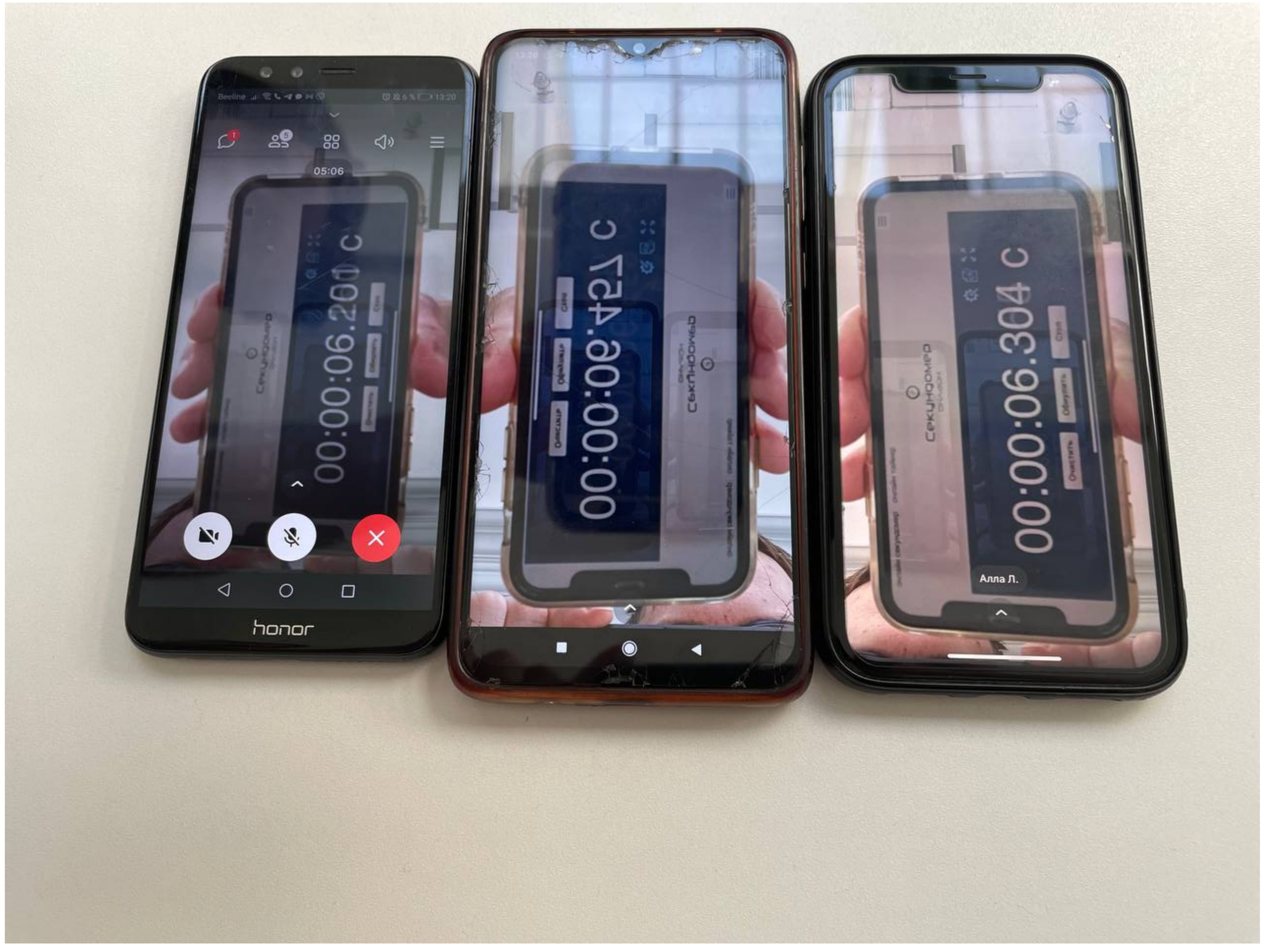
- отдача FIR с сервера
- медиана video passthrough **~40мс**
- 0,43 CPU на пользователя:
  - 0,34 CPU на видео
  - 0,09 CPU на аудио
- on-demand H264, VP8, VP9



Где мы находимся относительно конкурентов?

Оценка задержки видео







Результаты оценки задержки по видео

Desktop

VK (Web)	263
Skype (Desktop)	336
Google.Meet (Web)	350
VK (Native)	364
Zoom (Desktop)	457
FB MSNG (Web)	644

0 200 400 600

Desktop throttled

VK (Web)	260
VK (Native)	338
Skype (Desktop)	368
Google.Meet (Web)	413
FB MSNG (Web)	692
Zoom (Desktop)	793

0 200 400 600

Mobile

VK (Mob)	230
Google.Meet (Mob)	242
Zoom (Mob)	463
FB MSNG (Mob)	555
Skype (Mob)	583

0 200 400

- * Сеть выровнена по сервисам
- Мы везде лучшие — почему?



Синхронизация звука и видео

Lipsync

IS-191
G.1080
TR-126
IS-191
G.1080
TR-126
BT.1359 (acceptable)
BT.1359
BT.1359
BT.1359 (acceptable)
Steinmetz (annoying)
Steinmetz (acceptable)
R37
R37
Steinmetz (acceptable)
Steinmetz (annoying)
Curcio
Curcio
Ford
Ford
Younkin

-320 -300 -280 -260 -240 -220 -200 -180 -160 -140 -120 -100 -80 -60 -40 -20 0 20 40 60 80 100 120 140 160 180 200

Audio/Video delay (in ms)

audio delayed with respect to video — *audio ahead of video*

- отставание звука менее заметно, чем опережение
- не мешает [–100ms, +60ms]



Lipsync

- Не мешает **audio delay** [–100ms, +60ms]

	audio passthrough	**video** passthrough	**audio delay**
p50	60 ms	40 ms	–20 ms
p75	135 ms	80 ms	–55 ms
p99	500 ms	400 ms	–100 ms

- Не делаем lipsync на клиенте



Архитектура решения и 1000+

ZooKeeper
(Hosts Coordination)

Call info
Client info
Polling for transcoding Requests
Ttranscoding Requests

Alice
Bob
WebSocket
WebSocket

Signaler
Client Connections mgmt

Conferencing
Client Media Relay

Client media exchange

Transform
Client Media Transcoding

Media
Media

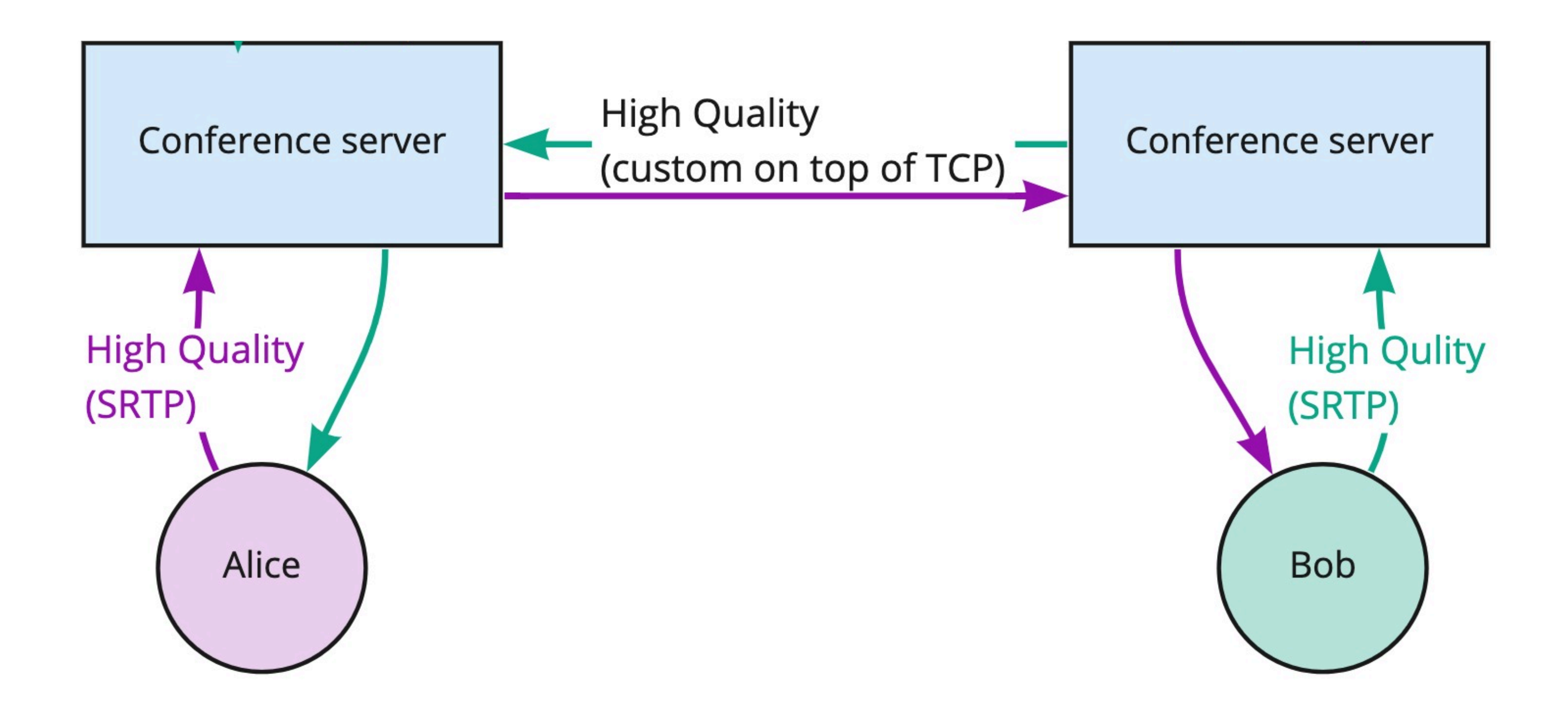
Host Load info

GSLB

- **Signaler**: websocket, Apache Tomcat, NFWare, Cassandra
- **Conferencing**: java, WebRTC, GSLBбалансировка
- **Transfrom <> Conferencing**: custom TCP

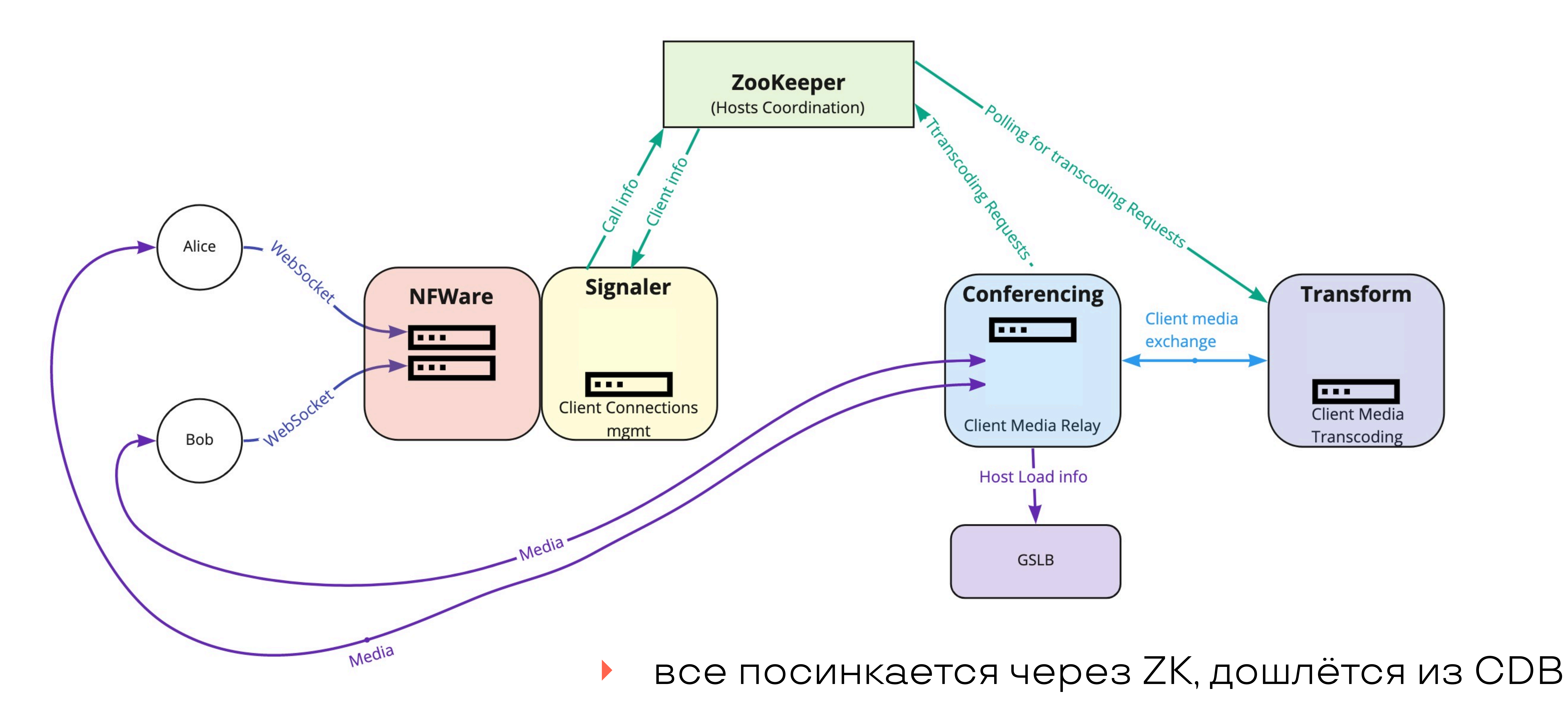


Скалирование одного звонка на любое N серверов





Отказоустойчивость



- все посинкается через ZK, дошлётся из CDB

Уменьшаем latency серверной архитектуры

- TCP_NODELAY

  WebRTC-сервер написан на **Java**
- Проблема: Safepoints / **GC** max **50ms**
- **Решение**: заменили дефолтный **G1GC** на **Shenandoah**, ориентированный на низкие паузы
- **Conferencing** max **10 мс**
- **Transform** max **3 мс**
- **Jitsi** тоже поможет)



Нельзя просто так взять и отправить на клиент 1000 видео

В WebRTC одно видео — один «трек» со стриминг-сервера до клиента:

- RTP «трек» — это пара SSRC.

Новый участник — дополнительная пара SSRC и переобмен SDP между клиентом и стриминг-сервером.

* Браузер умирает на 50 треках.

Server

Video stream 1
SDP
SSRC 1

Video stream 2
SDP
SSRC 2

Video stream N
SDP
SSRC N



Слоты для видеотреков

Server

Video stream 1

Video stream 2

Video stream k

Video stream n

Peer connection

SSRC 1

SSRC 2

SSRC 3

Маппинг стримов и треков



На 1000+ не обойтись без

- горизонтального скалирования звонка
- обхода ограничения на число треков в peer connection — слоты
- серверной топологии SFU/MIX
- NS и VAD



Улучшаем алгоритмы jb, bandwidth adaptation, time stretching, codec switching

Формальное решение задачи

$$f(x)$$

Многокритериальная оптимизация:

$$\min_{X,V}\{\textbf{задержка}(X,V),\ \textbf{искажения}(\mathrm{X},\mathrm{V}),\ \textbf{потребления ресурсов}(\mathrm{X},\mathrm{V})\},$$

где:

X = {BW, RTT, PL, Jitter, Device(CPU, OS, GPU, Resolution) | BROWSER, заряд батареи, перемещение в пространстве, количество участников, …}

***V* = {VIDEO SFU - QOD, onDemans{h264/vp8/vp9}, AUDIO MIX, Opus, V/NACK, A/FEC, CDN, VAD, NS V/A ADAPTATION TRICKS, TIME-STREATCHING}**



Модель, предсказывающая качество

signal_strength
amount_of_no_loader_freq
full_signal
avg_net_error_count
no_signal
mid_signal
avg_signal_strength
device_model
net_error_count
feed_first_data
city
asn_orgname
geo_hash
cold_inflate
os_version
avg_feed_first_data
median_net_error_count_week_ip

https://github.com/slundberg/shap



Тестирование на 1000+

ВКонТАКТЕ
Поиск
Павел
Моя страница
Назад
Autotests | Calls
Все сообщения
1:15:37 · 1024 участника
Говорят Михаил Т., Ян М., Руслан О.
Вы
Диана К.
Михаил Т.
Александр Т.
Ян М.
Иван Г.
Руслан О.
Яна К.
Илья С.
Наташа Т.
Филипп П.
Вадим Г.
Юлия К.
Александр Н.
Тема М.
Зара Б.
Иван Б.
Жора С.
Напишите сообщение...

- поднимаем браузеры на виртуалках



Готовые решения

Open-source media servers and SDKs

	Jitsi	Kurento	Mediasoup	Janus	Agora	VK SDK
Число участников S – speakers L – listeners	75 x 75	30 x 30	S x L = 500	10 + 140	17 + 1M	1000 x 1000
Архитектура	SFU + streaming	MCU / SFU	SFU	SFU / MCU plugins	SFU - 17 MCU - 1M	video SFU audio MCU
Технологии	**Java***, JavaScript	C/C++	Node.js, C++, TypeScript	C, C++, JavaScript	CDN	Java, C++

- Не забудьте потюнить GC*



Звонки своими руками

- Если вам **не** нужно бороться за **p99 и** не надо больше **100** участников

- Open source SFU (Jitsi, Mediasoup) будет работать для 30-50 человек
- Шумодав — Krisp

или

- Облачные решения (Agora, Voximplant) возьмут эксплуатацию и работу с операторами на себя



Итого

Что получилось и куда дальше

Цифры



6 млн/день

Звонки





Мы все разрабатываем удаленно



Функционал и конкуренты

	VK	SDK capacity	Viber	Fb	Discord	Skype	Zoom	Google Meet
Число участников	128	**video - 1000** **audio mix -1M**	30	50	50	50	100 ($ —1000)	100 ($ —250)
Платформы						*	*	
Подключение по ссылке	✓	✓	✓	✓	✗	✓	✓	✓
Демонстрация экрана	4K, High Res	4K, High Res/ High FPS	✓	✓	High FPS	✓	High Res	High FPS
Запись	✓	✓	✗	✗	✗	✓	локально	$$
Виртуальные фоны и маски	✓	✓	✗	✓	✗	✓	✓	✓



Метрики и конкуренты — крупно

	Zoom (Native)	Google Meet (Web)	VK SDK (Web)
audio delay	559 ms	350 ms	381 ms
video delay	457 ms	350 ms	263 ms
video throttle	793 ms	413 ms	309 ms
PL=50%	900 ms	max PL=10-20%	max PL=10-20%

- Zoom разменял задержку на работу с высоким PL и качество (требования пользователей*)
- Meet разменял задержку на работу с Web и разгрузку WebRTC-клиентов





Настоящее и будущее звонков

Как выглядят / будут выглядеть звонки

- ML VAD на градиентном бустинге
- ML NS на нейросетях
- ML audio codec — Google Lyra (3kbps)
- ML video codec
- ML улучшение изображения/лица
- ML оптимизация параметров алгоритмов адаптации — решающие деревья
- ML Оценка качества звонков — NISKA
- ML починка пропажи пакетов: PLC — 20мс, WaveNetEQ — 120мс

ML вытесняет эвристические алгоритмы



Общий подход к решению сложных инженерных задач

- **Определите требования**, сделайте опрос или коридорное исследование.
- Изучите **теорию** и **формализуйте** задачу.
- **Запустите** решение (можно на open-source).
- Соберите **метрики**.
- Определите, где вы находись **относительно** конкурентов.
- Развивайте, опираясь на эти знания и **метрики**.
- Заменяйте классические/эвристические алгоритмы на **ML**.



Главное понимать, где вы находитесь

Complexity

Zoom
Agora
Google Meet

OpenSource

Conference size
Low-latency
Video quality
...

СОН ДЛЯ
СЛАБАКОВ



Что мы вынесли сегодня



- сеть: BW, RTT, PL, Jitter
- починка сети и компенсация задержки: JB, NACK, FEC vs PLC
- audio pipeline: AEC, NS, VAD
- оценка качества: MOS, PESQ, NISKA
- топологиии : MCU, SFU, SVC, Simulcast, QOD
- DTX, FIR, I|P|B - frames
- Tricks и решения в разных зонах

ML +20–600%!



Что мне с этим делать?

- Встройте звонки в свой сервис.
- Пройдите собеседование в VOIP-команду.
- Начните свой стартап.



А если мне это не надо?

- Почти все эвристические алгоритмы заменил ML:
  - нейросети,
  - градиентный бустинг.
- Это общая тенденция, мы её рассмотрели на VOIP.
- В вашем проекте точно есть, где улучшить!



ВНЕДРЯЙ ML!

Будем ВКонтакте!



Александр Тоболь, CTO ВКонтакте

alexander.tobol@corp.vk.com

https://vk.com/alatobol